

```
*****
* CBIOS FOR CP/M VER 2.2 FOR DISK JOCKEY 2D CONTROLLER (ALL
* REVS, AND MODELS A & B). HANDLES DISKETTES WITH SECTOR SIZES
* OF 128 BYTES SINGLE DENSITY, 256, 512, 1024 BYTES DOUBLE
* DENSITY. THERE ARE CONDITIONAL ASSEMBLIES FOR DISKUS HARD
* DISK CONTROLLER.
*
* WRITTEN BY BOBBY DALE GIFFORD.
* 12/8/80
*
* CUSTOMIZED BY JAY O'BRIEN
* 1/16/82
*
* DISK MAP OF SECTORS USED BY COLD BOOT, WARM BOOT, FIRMWARE,
* AND CP/M:
*
* TRK 0 SEC 1 = FIRST SECTOR OF COLD BOOT. E700H
* 0 2 = COLD BOOT 256. 80H
* 0 3 = COLD BOOT 512. 80H
* 0 4 = COLD BOOT 1024. 80H
* 0 5 = WARM BOOT 256. 80H
* 0 6 = WARM BOOT 512. 80H
* 0 7 = WARM BOOT 1024. 80H
* 0 8 = COLD/WARM BOOT. 2C00H
* 0 9 = FIRMWARE. E400H
* 0 10 = FIRMWARE+80H. E480H
* 0 11 = FIRMWARE+100H. E500H
* 0 12 = FIRMWARE+180H. E580H
* 0 13 = FIRMWARE+200H. E600H
* 0 14 = FIRMWARE+280H. E680H
* 0 15 = FIRMWARE+300H. E700H
* 0 16 = FIRMWARE+380H. E780H
* 0 17 = CCP. 2700H
* 0 18 = CCP+80H. 2780H
* 0 19 = CCP+100H. 2800H
* 0 20 = CCP+180H. 2880H
* 0 21 = CCP+200H. 2900H
* 0 22 = CCP+280H. 2980H
* 0 23 = CCP+300H. 2A00H
* 0 24 = CCP+380H. 2A80H
* 0 25 = CCP+400H. 2B00H
* 0 26 = CCP+480H. 2B80H
* 1 = REST OF CP/M. 2C00H-4FFFH
```

TITLE '\*\*\* Cbios For CP/M Ver. 2.2 \*\*\*'

\* THE FOLLOWING REVISION NUMBER IS IN REFERENCE TO THE CP/  
\* 2.2 CBIOS.

CB1057A.PRN

3/23/82

DJ AT FOCO

FLOPPIES=A, B  
HD=C, D, E

V10X VIDEO

60K

```

001C =      REVNUM EQU 28      ;CBIOS REVISION NUMBER
0016 =      CPMREV EQU 22      ;CP/M REVISION NUMBER

*****
* THE FOLLOWING EQUATES SET UP THE RELATIONSHIP BETWEEN THE
* 2D FLOPPIES AND THE HARD DISK CONTROLLERS.
*

*****
0000 =      FIRST  EQU 0      ;0 = FLOPPIES ARE A,B,C,D DRIVES AND
;          ; HARD DISK ARE E,F,G,H
;          ;1 = HARD DISKS ARE A,B,C,D DRIVES AND
;          ; FLOPPIES ARE E,F,G,H
0001 =      MAXHD  EQU 1      ;SET TO NUMBER OF HARD DISKS
0002 =      MAXFLOP EQU 2      ;SET TO NUMBER OF FLOPPIES

0001 =      M26   EQU 1      ;SET ONLY ONE OF THESE VARIABLES
0000 =      M20   EQU 0
0000 =      M10   EQU 0

          IF      M10 OR M20
          SDELAY EQU 0      ;SOFTWARE HEAD SETTLE DELAY (0 = NO, 1 = YES)
          ELSE
          SDELAY EQU 1
          ENDIF

001A =      MREV  EQU 26*M26+20*M20+10*M10      ;HARD DISK TYPE
0003 =      LOGDSK EQU 3*M26+3*M20+2*M10      ;LOGICAL DISKS PER DRIVE
0020 =      HDSPT EQU 32*M26+21*M20+21*M10      ;SECTORS PER TRACK

*****
* THE FOLLOWING EQUATES RELATE THE THINKER TOYS 2D CONTROLLER.
* IF THE CONTROLLER IS NON STANDARD (0E000H) ONLY THE ORIGIN
* EQUATE NEED BE CHANGED. THIS VERSION OF THE CBIOS WILL WORK
* WITH 2D CONTROLLER BOARDS REV 0, 1, 3, 3.1, 4, MODEL B.
*
*****
```

F000 =	ORIGIN EQU	IF MAXFLOP NE 0	;INCLUDE DISCUS 2D ?
F400 =	DJRAM EQU	0F000H	;DISK JOCKEY 2D RAM ADDRESS
F400 =	DJBOOT EQU	DJRAM	;DISK JOCKEY 2D INITIALIZATION
F003 =	DJCIN EQU	ORIGIN+3H	;DISK JOCKEY 2D CHARACTER INPUT ROUTINE
F006 =	DJCOUT EQU	ORIGIN+6H	;DISK JOCKEY 2D CHARACTER OUTPUT ROUTINE
F409 =	DJHOME EQU	DJRAM+9H	;DISK JOCKEY 2D TRACK ZERO SEEK
F40C =	DJTRK EQU	DJRAM+0CH	;DISK JOCKEY 2D TRACK SEEK ROUTINE
F40F =	DJSEC EQU	DJRAM+0FH	;DISK JOCKEY 2D SET SECTOR ROUTINE
F412 =	DJDMA EQU	DJRAM+012H	;DISK JOCKEY 2D SET DMA ADDRESS
F415 =	DJREAD EQU	DJRAM+15H	;DISK JOCKEY 2D READ ROUTINE
F418 =	DJWRITE EQU	DJRAM+18H	;DISK JOCKEY 2D WRITE ROUTINE
F41B =	DJSEL EQU	DJRAM+1BH	;DISK JOCKEY 2D SELECT DRIVE ROUTINE
F021 =	DJTSTAT EQU	ORIGIN+21H	;DISK JOCKEY 2D TERMINAL STATUS ROUTINE

```

F427 =      DJSTAT EQU  DJRAM+27H ;DISK JOCKEY 2D STATUS ROUTINE
F42A =      DJERR  EQU  DJRAM+2AH ;DISK JOCKEY 2D ERROR, FLASH LED
F42D =      DJDEN   EQU  DJRAM+2DH ;DISK JOCKEY 2D SET DENSITY ROUTINE
F430 =      DJSIDE  EQU  DJRAM+30H ;DISK JOCKEY 2D SET SIDE ROUTINE
0008 =      DBLSID EQU  8      ;SIDE BIT FROM CONTROLLER
            ENDIF

```

```

*****
* THE FOLLOWING EQUATES ARE FOR THE DISKUS HARD DISK WANTED.
*****

```

```

0050 =      IF      MAXHD NE 0      ;WANT HARD DISK INCLUDED ?
0050 =      HDORG  EQU  50H      ;HARD DISK CONTROLLER ORIGIN
0050 =      HDSTAT  EQU  HDORG    ;HARD DISK STATUS
0050 =      HDCNTL  EQU  HDORG    ;HARD DISK CONTROL
0053 =      HDDATA  EQU  HDORG+3  ;HARD DISK DATA
0052 =      HDFUNC  EQU  HDORG+2  ;HARD DISK FUNCTION
0051 =      HDMCMD  EQU  HDORG+1  ;HARD DISK COMMAND
0051 =      HDRESLT EQU  HDORG+1  ;HARD DISK RESULT
0002 =      RETRY   EQU  2      ;RETRY BIT OF RESULT
0001 =      TKZERO  EQU  1      ;TRACK ZERO BIT OF STATUS
0002 =      OPDONE   EQU  2      ;OPERACTION DONE BIT OF STATUS
0004 =      COMPLT  EQU  4      ;COMPLETE BIT OF STATUS
0008 =      TMOUT   EQU  8      ;TIME OUT BIT OF STATUS
0010 =      WFAULT  EQU  10H    ;WRITE FAULT BIT OF STATUS
0020 =      DRVRDY  EQU  20H    ;DRIVE READY BIT OF STATUS
0040 =      INDEX   EQU  40H    ;INDEX BIT OF STATUS
0004 =      PSTEP   EQU  4      ;STEP BIT OF FUNCTION
00FB =      NSTEP   EQU  0FBH   ;STEP BIT MASK OF FUNCTION
0004 =      HDRLEN  EQU  4      ;SECTOR HEADER LENGTH
0200 =      SECLEN  EQU  512    ;SECTOR DATA LENGTH
000F =      WENABL  EQU  0FH    ;WRITE ENABLE
000B =      WRESET  EQU  0BH    ;WRITE RESET OF FUNCTION
0005 =      SCENBL  EQU  5      ;CONTROLLER CONTROL
0007 =      DSKCLK  EQU  7      ;DISK CLOCK FOR CONTROL
00F7 =      MDIR    EQU  0F7H   ;DIRECTION MASK FOR FUNCTION
00FC =      NULL    EQU  0FCH   ;NULL COMMAND
0000 =      IDBUFF  EQU  0      ;INITIALIZE DATA COMMAND
0008 =      ISBUFF  EQU  8      ;INITIALIZE HEADER COMMAND
0001 =      RSECT   EQU  1      ;READ SECTOR COMMAND
0005 =      WSECT   EQU  5      ;WRITE SECTOR COMMAND
            ENDIF

```

```

*****
* CP/M SYSTEM EQUATES. IF RECONFIGURATION OF THE CP/M SYSTEM
* IS BEING DONE, THE CHANGES CAN BE MADE TO THE FOLLOWING
* EQUATES.
*****

```

```

003C =      MSIZE  EQU  60      ;MEMORY SIZE OF TARGET CP/M
A000 =      BIAS    EQU  (MSIZE-20)*1024 ;MEMORY OFFSET FROM 20K SYSTEM
C700 =      CCP     EQU  2700H+BIAS ;CONSOLE COMMAND PROCESSOR

```

CF00 =	BDOS	EQU	CCP+800H	;BDOS ADDRESS
DD00 =	BIOS	EQU	CCP+1600H	;CBIOS ADDRESS
4A00 =	OFFSETC	EQU	2700H-BIOS	;OFFSET FOR SYSGEN
0004 =	CDISK	EQU	4	;ADDRESS OF LAST LOGGED DISK
0080 =	BUFF	EQU	80H	;DEFAULT BUFFER ADDRESS
0100 =	TPA	EQU	100H	;TRANSIENT MEMORY
00C0 =	INTIOBY	EQU	192	;INITIAL IOBYTE
0003 =	IOBYTE	EQU	3	;IOBYTE LOCATION
0000 =	WBOT	EQU	0	;WARM BOOT JUMP ADDRESS
0005 =	ENTRY	EQU	5	;BDOS ENTRY JUMP ADDRESS

\*\*\*\*\*  
\* \*  
\* THE FOLLOWING ARE INTERNAL CBIOS EQUATES. MOST ARE MISC.  
\* CONSTANTS.  
\* \*  
\*\*\*\*\*

000A =	RETRIES	EQU	10	;MAX RETRIES ON DISK I/O BEFORE ERROR
000D =	ACR	EQU	0DH	;A CARRIAGE RETURN
000A =	ALF	EQU	0AH	;A LINE FEED
001A =	CLEAR	EQU	1AH	;CLEAR SCREEN FOR VIO-X
0003 =	AETX	EQU	3	;ETX CHARACTER
0006 =	AACK	EQU	6	;ACK CHARACTER

\*\*\*\*\*  
\* \*  
\* THE JUMP TABLE BELOW MUST REMAIN IN THE SAME ORDER, THE  
\* ROUTINES MAY BE CHANGED, BUT THE FUNCTION EXECUTED MUST BE  
\* THE SAME.  
\* \*  
\*\*\*\*\*

		ORG	BIOS	;CBIOS STARTING ADDRESS
DD00 C3DAE5		JMP	CBOOT	;COLD BOOT ENTRY POINT
DD03 C3BCDE	WBOOTE	JMP	WBOOT	;WARM BOOT ENTRY POINT
DD06 C336DD		JMP	CONST	;CONSOLE STATUS ROUTINE
DD09 C342DD		JMP	CONIN	;CONSOLE INPUT
DD0C C357DD	COUT	JMP	CONOUT	;CONSOLE OUTPUT
DD0F C377DD		JMP	LIST	;LIST DEVICE OUTPUT
DD12 C36CDD		JMP	PUNCH	;PUNCH DEVICE OUTPUT
DD15 C362DD		JMP	READER	;READER DEVICE INPUT
DD18 C351DF		JMP	HOME	;HOME DRIVE
DD1B C393DF		JMP	SETDRV	;SELECT DISK
DD1E C353DF		JMP	SETTRK	;SET TRACK
DD21 C345DF		JMP	SETSEC	;SET SECTOR
DD24 C34BDF		JMP	SETDMA	;SET DMA ADDRESS
DD27 C399E0		JMP	READ	;READ THE DISK
DD2A C392E0		JMP	WRITE	;WRITE THE DISK
DD2D C382DD		JMP	LISTST	;LIST DEVICE STATUS
DD30 C358DF		JMP	SECTRAN	;SECTOR TRANSLATION
DD33 C31BF4	DJDRV	IF	MAXFLOP NE 0	
		JMP	DJSEL	;HOOK FOR SINGLE.COM PROGRAM
		ELSE		

JMP DONO  
ENDIF

```
*****
* TERMINAL DRIVER ROUTINES. IOBYTE IS INITIALIZED BY THE COLD
* BOOT ROUTINE, TO MODIFY, CHANGE THE "INTIOBY" EQUATE. THE
* I/O ROUTINES THAT FOLLOW ALL WORK EXACTLY THE SAME WAY. USING
* IOBYTE, THEY OBTAIN THE ADDRESS TO JUMP TO IN ORDER TO EXECUTE
* THE DESIRED FUNCTION. THERE IS A TABLE WITH FOUR ENTRIES FOR
* EACH OF THE POSSIBLE ASSIGNMENTS FOR EACH DEVICE. TO MODIFY
* THE I/O ROUTINES FOR A DIFFERENT I/O CONFIGURATION, JUST
* CHANGE THE ENTRIES IN THE TABLES.
*
```

F003 = CITY EQU DJCIN ; INPUT FROM THE DISK JOCKEY 2  
F006 = COTTY EQU DJCOUT ; OUTPUT TO THE DISK JOCKEY 2D

```
*****
* CONST: GET THE STATUS FOR THE CURRENTLY ASSIGNED CONSOLE
* DEVICE. THE CONSOLE DEVICE CAN BE GOTTEN FROM IOBYTE,
* THEN A JUMP TO THE CORRECT CONSOLE STATUS ROUTINE IS
* PERFORMED.
*****
```

DD36 21BØDD CONST LXI H,CSTBLE ;BEGINNING OF JUMP TABL  
DD39 C348DD JMP CONIN1 ;SELECT CORRECT JUMP

```
*****
* CSREADER: IF THE CONSOLE IS ASSIGNED TO THE READER THEN A
* JUMP WILL BE MADE HERE, WHERE ANOTHER JUMP WILL
* OCCUR TO THE CORRECT READER STATUS.
*****
```

DD3C 21B8DD CSREADR LXI H,CSRTBLE ;BEGINNING OF READER STATUS TABL  
DD3F C365DD JMP READERA

```
*****
* CONIN: TAKE THE CORRECT JUMP FOR THE CONSOLE INPUT ROUTINE.
* THE JUMP IS BASED ON THE TWO LEAST SIGNIFICANT BITS OF
* IOBYTE.
```

DD42 CD0CE1 CONIN CALL FLUSH ;FLUSH THE DISK BUFFER  
DD45 2188DD LXI H,CITBLE ;BEGINNING OF CHARACTER INPUT TABLE

\*  
\* ENTRY AT CONINI WILL DECODE THE TWO LEAST SIGNIFICANT BITS  
\* OF IOBYTE. THIS IS USED BY CONIN, CONOUT, AND CONST.

\*

DD48 3A0300 CONIN1 LDA IOBYTE  
DD4B 17 RAL

\*

\* ENTRY AT SELDEV WILL FORM AN OFFSET INTO THE TABLE POINTED  
\* TO BY H&L AND THEN PICK UP THE ADDRESS AND JUMP THERE.  
\*

DD4C E606 SELDEV ANI 6H ;STRIP OFF UNWANTED BITS  
DD4E 1600 MVI D,0 ;FORM OFFSET  
DD50 5F MOV E,A  
DD51 19 DAD D ;ADD OFFSET  
DD52 7E MOV A,M ;PICK UP HIGH BYTE  
DD53 23 INX H  
DD54 66 MOV H,M ;PICK UP LOW BYTE  
DD55 6F MOV L,A ;FORM ADDRESS  
DD56 E9 PCHL ;GO THERE !

\*\*\*\*\*

\*  
\* CONOUT: TAKE THE PROPER BRANCH ADDRESS BASED ON THE TWO LEAST \*  
\* SIGNIFICANT BITS OF IOBYTE.  
\*

\*\*\*\*\*

DD57 C5 CONOUT PUSH B ;SAVE THE CHARACTER  
DD58 CD0CE1 CALL FLUSH ;FLUSH THE DISK BUFFER  
DD5B C1 POP B ;RESTORE THE CHARACTER  
DD5C 2190DD LXI H,COTBLE ;BEGINNING OF THE CHARACTER OUT TABLE  
DD5F C348DD JMP CONINI ;DO THE DECODE

\*\*\*\*\*

\*  
\* READER: SELECT THE CORRECT READER DEVICE FOR INPUT. THE \*  
\* READER IS SELECTED FROM BITS 2 AND 3 OF IOBYTE.  
\*

\*\*\*\*\*

DD62 21A8DD READER LXI H,RTBLE ;BEGINNING OF READER INPUT TABLE

\*  
\* ENTRY AT READERA WILL DECODE BITS 2 & 3 OF IOBYTE, USED  
\* BY CSREADER.  
\*

DD65 3A0300 READERA LDA IOBYTE

\*  
\* ENTRY AT READER1 WILL SHIFT THE BITS INTO POSITION, USED  
\* BY LIST AND PUNCH.  
\*

DD68 1F READER1 RAR  
DD69 C34CDD JMP SELDEV

```
*****
* PUNCH: SELECT THE CORRECT PUNCH DEVICE. THE SELECTION COMES
* FROM BITS 4&5 OF IOBYTE.
*****
```

```
DD6C 21A0DD PUNCH LXI H,PTBLE ;BEGINNING OF PUNCH TABLE
DD6F 3A0300 LDA IOBYTE
```

```
*
* ENTRY AT PNCH1 ROTATES BITS A LITTLE MORE IN PREP FOR
* SELDEV, USED BY LIST.
*
```

```
DD72 1F PNCH1 RAR
DD73 1F RAR
DD74 C368DD JMP READR1
```

```
*****
* LIST: SELECT A LIST DEVICE BASED ON BITS 6&7 OF IOBYTE
*****
```

```
DD77 2198DD LIST LXI H,LTBLE ;BEGINNING OF THE LIST DEVICE ROUTINES
DD7A 3A0300 LIST1 LDA IOBYTE
DD7D 1F RAR
DD7E 1F RAR
DD7F C372DD JMP PNCH1
```

```
*****
* LISTST: GET THE STATUS OF THE CURRENTLY ASSIGNED LIST DEVICE
*****
```

```
DD82 21C0DD LISTST LXI H,LSTBLE ;BEGINNING OF THE LIST DEVICE STATUS
DD85 C37ADD JMP LIST1
```

```
*****
* IF CUSTOMIZING I/O ROUTINES IS BEING PERFORMED, THE TABLE
* BELOW SHOULD BE MODIFIED TO REFLECT THE CHANGES. ALL I/O
* DEVICES ARE DECODED OUT OF IOBYTE AND THE JUMP IS TAKEN FROM
* THE FOLLOWING TABLES.
*****
```

```
*****
* CONSOLE INPUT TABLE
*****
```

```
DD88 F3DD CITBLE DW CIUC1 ; INPUT FROM USER CONSOLE 1 (CURRENTLY
; SWBD PARALLEL PORT 4)
```

CP/M MACRO ASSEM 2.0 #008 \*\*\* Cbios For CP/M Ver. 2.2 \*\*\*

DD8A 08DE DW CICRT ;INPUT FROM CRT (CURRENTLY SWITCHBOARD SERIAL PORT 1)  
 DD8C 62DD DW READER ;INPUT FROM READER (DEPENDS ON READER SELECTION)  
 DD8E 03F0 DW CITY ;INPUT FROM TTY (CURRENTLY INPUT FROM DISK JOCKEY 2D)

\*

\* CONSOLE OUTPUT TABLE

DD90 3BDE COTBLE DW COCRT ;OUTPUT TO CRT  
 DD92 3BDE DW COCRT ;OUTPUT TO CRT  
 DD94 77DD DW LIST ;OUTPUT TO LIST DEVICE (DEPENDS ON BITS 6&7 OF IOBYTE)  
 DD96 06F0 DW COTTY ;OUTPUT TO TTY (CURRENTLY OUTPUT TO DISK JOCKEY 2D)

\*

\* LIST DEVICE TABLE

DD98 06F0 LTBLE DW COTTY ;OUTPUT TO TTY (CURRENTLY ASSIGNED BY INTIOBY, OUTPUT TO 2D)  
 DD9A 46DE DW COPTR ;OUTPUT TO PRINTER *OK!*  
 DD9C C9DD DW COLPT ;OUTPUT TO LINE PRINTER (CURRENTLY SWITCHBOARD SERIAL PORT 1)  
 DD9E D4DD DW COUL1 ;OUTPUT TO USER LINE PRINTER 1 (CURRENTLY *1PS1* SWITCHBOARD SERIAL PORT 1)

\*

\* PUNCH DEVICE TABLE

DDA0 06F0 PTBLE DW COTTY ;OUTPUT TO THE TTY (CURRENTLY ASSIGNED BY INTIOBY, OUTPUT TO 2D)  
 DDA2 46DE DW COPTR ;OUTPUT TO PRINTER  
 DDA4 C9DD DW COUP1 ;OUTPUT TO USER PUNCH 1 (CURRENTLY SWITCHBOARD SERIAL PORT 1)  
 DDA6 C9DD DW COUP2 ;OUTPUT TO USER PUNCH 2 (CURRENTLY SWITCHBOARD SERIAL PORT 1)

\*

\* READER DEVICE INPUT TABLE

DDA8 03F0 RTBLE DW CITY ;INPUT FROM TTY (CURRENTLY ASSIGNED BY INTIOBY, INPUT FROM 2D)  
 DDAA 08DE DW CIPTR ;INPUT FROM PAPER TAPE READER (CURRENTLY SWITCHBOARD SERIAL PORT 1)  
 DDAC 08DE DW CIUR1 ;INPUT FROM USER READER 1 (CURRENTLY SWITCHBOARD SERIAL PORT 1)  
 DDAE 08DE DW CIUR2 ;INPUT FROM USER READER 2 (CURRENTLY

; SWITCHBOARD SERIAL PORT 1)

\*  
\* CONSOLE STATUS TABLE  
\*

DDB0 FFDD	CSTBLE DW	CSUC1	; STATUS FROM SWBD PARALLEL PORT 4, AS ; READ FROM ATTN BIT 0)
DDB2 1CDE	DW	CSCRT	; STATUS FROM CRT (CURRENTLY SWITCHBOARD ; SERIAL PORT 1)
DDB4 3CDD	DW	CSREADR	; STATUS FROM READER (DEPENDS ON READER DEVICE )
DDB6 14DE	DW	CSTTY	; STATUS OF TTY (CURRENTLY STSTUS FROM ; DISK JOCKEY 2D)

\*  
\* STATUS FROM READER DEVICE  
\*

DDB8 14DE	CSRTBLE DW	CSTTY	; STATUS FROM TTY (CURRENTLY ASSIGNED ; BY INTIOBY, STATUS OF 2D)
DDBA 1CDE	DW	CSPTR	; STATUS FROM PAPER TAPE READER (CURRENTLY ; SWITCHBOARD SERIAL PORT 1)
DDBC 1CDE	DW	CSUR1	; STATUS FROM USER READER 1 (CURRENTLY ; SWITCHBOARD SERIAL PORT 1)
DBBE 1CDE	DW	CSUR2	; STATUS OF USER READER 2 (CURRENTLY ; SWITCHBOARD SERIAL PORT 1)

\*  
\* STATUS FROM LIST DEVICE  
\*

DDC0 2ADE	LSTBLE DW	READY	; CONSOLE ALWAYS READY
DDC2 2ADE	DW	READY	; GET LIST STATUS
DDC4 25DE	DW	LSLPT	
DDC6 25DE	DW	LSLPT	

\*\*\*\*\*  
\*  
\* ROUTINES FOR MY SYSTEM. J. J. O'BRIEN  
\*  
\*\*\*\*\*  
\*  
\* THE FOLLOWING EQUATES SET OUTPUT DEVICE TO OUTPUT TO THE  
\* SWITCHBOARD SERIAL PORT 1.  
\*  
\*\*\*\*\*

DDC9 =	COPTP EQU	\$	; OUTPUT FROM PAPER TAPE PUNCH
DDC9 =	COUP1 EQU	\$	; OUTPUT FROM USER PUNCH 1
DDC9 =	COUP2 EQU	\$	; OUTPUT FROM USER PUNCH 2
DDC9 DB02	COLPT IN	2	; OUTPUT FROM LINE PRINTER,GET STATUS
DDCB E680	ANI 80H		;WAIT UNTIL OK TO SEND
DDCD CAC9DD	JZ COLPT		
DDD0 79	MOV A,C		;OUTPUT THE CHARACTER

DDD1 D301 OUT 1  
 DDD3 C9 RET

\*\*\*\*\*  
 \*  
 \* CUSTOM I/O PRINTER DRIVER FOR DIABLO PRINTER WITH 1200 BAUD \*  
 \* ETX/ACK HANDSHAKE.  
 \*  
 \*\*\*\*\*

DDD4 CDC9DD	COUL1	CALL	COLPT	; OUTPUT THE CHARACTER
DDD7 3AF2DD		LDA	COUNT	
DDDA 3D		DCR	A	
DDDB 32F2DD		STA	COUNT	
DDDE C0		RNZ		
DDDF 3E4E		MVI	A, 78	
DDE1 32F2DD		STA	COUNT	
DDE4 0E03		MVI	C,AETX	
DDE6 CDC9DD		CALL	COLPT	
DDE9 CD08DE	PWAIT	CALL	CIPTR	
DDEC FE06		CPI	AACK	
DDEE C2E9DD		JNZ	PWAIT	
DDF1 C9		RET		

DDF2 32 COUNT DB 50

\*\*\*\*\*  
 \*  
 \* THE FOLLOWING EQUATES SET THE INPUT TO COME FROM THE SWBD \*  
 \* PARALLEL PORT 4, WITH STATUS ON ATTENTION PORT BIT 0.  
 \*  
 \*\*\*\*\*

DDF3 DB03	CIUC1	IN	3	; GET ATTENTION BYTE
DDF5 E601		ANI	1	; GET BIT 0 ONLY
DDF7 CAF3DD		JZ	CIUC1	; WAIT FOR CHARACTER
DDFA DB04		IN	4	; GET CHARACTER
DDFC E67F		ANI	7FH	; STRIP OFF THE PARITY
DDFE C9		RET		

DDFF DB03	CSUC1	IN	3	; GET ATTENTION BYTE
DE01 E601		ANI	1	; GET BIT 0 ONLY
DE03 EE01		XRI	1	; CHANGE POLARITY
DE05 C317DE		JMP	STAT	; RETURN PROPER INDICATION

\*\*\*\*\*  
 \*  
 \* THE FOLLOWING EQUATES SET THE INPUT FROM THE DEVICES TO COME \*  
 \* FROM THE SWITCHBOARD SERIAL PORT 1.  
 \*  
 \*\*\*\*\*

DE08 =	CICRT	EQU	\$	; INPUT FROM CRT
DE08 =	CIUR1	EQU	\$	; INPUT FROM USER READER 1
DE08 =	CIUR2	EQU	\$	; INPUT FROM USER READER 2
DE08 DB02	CIPTR	IN	2	; INPUT FROM PAPER TAPE READER, GET STATUS

```
DE0A E640      ANI    40H      ;WAIT FOR CHARACTER
DE0C CA08DE    JZ     CIPTR
DE0F DB01      IN     1
DE11 E67F      ANI    7FH      ;STRIP OFF THE PARITY
DE13 C9        RET
```

```
*****
* CONSOLE STATUS ROUTINES, TEST IF A CHARACTER HAS ARRIVED.
*****
*****
```

```
DE14 CD21F0    CSTTY  CALL    DJTSTAT    ;STATUS FROM DISK JOCKEY 2D
DE17 3E00      STAT   MVI     A,0      ;PREP FOR ZERO RETURN
DE19 C0        RNZ    A
DE1A 3D        DCR    A      ;RETURN WITH 0FFH
DE1B C9        RET
```

```
*****
* THE FOLLOWING EQUATES CAUSE THE DEVICES TO GET STATUS FROM
* THE SWITCHBOARD SERIAL PORT 1.
*****
*****
```

```
DE1C =        CSUR1  EQU    $      ;STATUS OF USER READER 1
DE1C =        CSUR2  EQU    $      ;STATUS OF USER READER 2
DE1C =        CSPTR  EQU    $      ;STATUS OF PAPER TAPE READER
DE1C DB02      CSCRT  IN     2      ;STATUS FROM CRT, GET STATUS
DE1E E640      ANI    40H      ;STRIP OF DATA READY BIT
DE20 EE40      XRI    40H      ;MAKE CORRECT POLARITY
DE22 C317DE    JMP    STAT    ;RETURN PROPER INDICATION
```

```
*****
* LIST DEVICE STATUS ROUTINES.
*****
*****
```

```
DE25 DB02      LSLPT  IN     2      ;ALL OTHER DEVICES WAIT
DE27 E680      ANI    80H
DE29 C8        RZ
DE2A 3EFF      READY  MVI    A,0FFH
DE2C C9        RET
```

```
*****
* THIS INITIALIZING ROUTINE SAMPLES BIT 0 OF SWBD PORT 7 TO
* DETERMINE IF THE KEYBOARD IS PLUGGED IN.  IF THE KEYBOARD IS
* PLUGGED IN, THE LSB RETURNS A 0. OTHERWISE, IT IS A 1.
* THIS 1 IS ADDED TO IOBYTE TO CHANGE THE CONSOLE INPUT FROM
* THE SWBD PARALLEL PORT 4 (THE KEYBOARD) TO THE SWBD SERIAL
* PORT THAT RECEIVES RS232 DATA FROM THE RS232 TERMINAL.
*****
*****
```

```

DE2D 0E1A      TINIT    MVI     C,CLEAR      ;INITIALIZE THE TERMINAL ROUTINE
DE2F DB07      IN      7          ;GET KEYBOARD INTERLOCK BYTE
DE31 E601      ANI     1          ;GET BIT 1 ONLY
DE33 C6C0      ADI     INTIOBY    ;ADD INTIOBY TO KEYBOARD BIT
DE35 320300    STA     IOBYTE     ;INITIALIZE IOBYTE
DE38 C30CDD    JMP     COUT      ;COUT

```

```

*****
*          *
*  VIO-X VIDEO DRIVER
*          *
*****

```

```

DE3B DB09      COCRT   IN      9          ;READ STATUS PORT
DE3D E601      ANI     1          ;MASK TXRDY BIT
DE3F CA3BDE    JZ      COCRT     ;WAIT FOR READY
DE42 79        MOV     A,C       ;GET CHAR
DE43 D308      OUT    8          ;OUTPUT IT
DE45 C9        RET

```

```

*****
*          *
*  ROUTINE FOR OKIDATA PRINTER
*  PRINTER IS ON PORT 0 WITH PRINTER READY ON PORT 5 BIT 1
*          *
*****

```

```

DE46 DB02      COPTR   IN      2          ;INPUT FROM PORT 2
DE48 E608      ANI     8          ;WAIT UNTIL OK TO SEND
DE4A CA46DE    JZ      COPTR     ;COPTR
DE4D DB05      COPTR1  IN      5          ;BUFFER FULL?
DE4F E601      ANI     1          ;COPTR1
DE51 CA4DDE    JZ      COPTR1    ;WAIT UNTIL PRINTER READY
DE54 79        MOV     A,C       ;OUTPUT THE CHARACTER
DE55 D300      OUT    0          ;COPTR1
DE57 C9        RET

```

```

*****
*          *
*  GOCPM IS THE ENTRY POINT FROM COLD BOOTS, AND WARM BOOTS. IT
*  INITIALIZES SOME OF THE LOCATIONS IN PAGE 0, AND SETS UP THE
*  INITIAL DMA ADDRESS (80H).
*          *
*****

```

```

DE58 218000    GOCPM   LXI     H,BUFF      ;SET UP INITIAL DMA ADDRESS
DE5B CD4BDF    CALL    SETDMA    ;INITIALIZE JUMP TO WARM BOOT
DE5E 3EC3      MVI     A,(JMP)    ;INITIALIZE JUMP TO BDOS
DE60 320000    STA     WBOT      ;INITIALIZE JUMP TO BDOS
DE63 320500    STA     ENTRY     ;ADDRESS IN WARM BOOT JUMP
DE66 2103DD    LXI     H,WBOOTE   ;ADDRESS IN BDOS JUMP
DE69 220100    SHLD    WBOT+1    ;ADDRESS IN BDOS JUMP
DE6C 2106CF    LXI     H,BDOS+6   ;ADDRESS IN BDOS JUMP
DE6F 220600    SHLD    ENTRY+1    ;A <- 0
DE72 AF        XRA     A
DE73 3205E5    STA     BUFSEC    ;DISK JOCKEY BUFFER EMPTY

```

DE76 320DE1	STA	BUFWRTN	:SET BUFFER NOT DIRTY FLAG
DE79 3A0400	LDA	CDISK	:JUMP TO CP/M WITH CURRENTLY SELECTED DISK IN C
DE7C 4F	MOV	C,A	
DE7D 3AAADE	LDA	CWFLG	
DE80 A7	ANA	A	
DE81 11ACDE	LXI	D,COLDBEG	:BEGINNING OF INITIAL COMMAND
DE84 3E0F	MVI	A,COLDEND-COLDBEG+1	;LENGTH OF COMMAND
DE86 CA8EDE	JZ	CLDCMND	
DE89 11BBDE	LXI	D,WARMBEG	
DE8C 3E01	MVI	A,WARMEND-WARMBEG+1	
DE8E 2108C7	CLDCMND	LXI	H,CCP+8 ;COMMAND BUFFER
DE91 3207C7	STA	CCP+7	
DE94 47	MOV	B,A	
DE95 CDD4E1	CALL	MOVLOP	
DE98 3AAADE	LDA	CWFLG	
DE9B A7	ANA	A	
DE9C 3AABDE	LDA	AUTOFLG	
DE9F CAA3DE	JZ	CLDBOT	
DEA2 1F	RAR		
DEA3 1F	CLDBOT	RAR	
DEA4 DA00C7	JC	CCP	
DEA7 C303C7	JMP	CCP+3 ;ENTER CP/M	
DEAA 00	CWFLG	DB 0	;COLD/WARM BOOT FLAG

\*\*\*\*\*
\* THE FOLLOWING BYTE DETERMINES IF AN INITIAL COMMAND IS TO BE
\* GIVEN TO CP/M ON WARM OR COLD BOOTS. THE VALUE OF THE BYTE IS
\* USED TO GIVE THE COMMAND TO CP/M:
\*
\* 0 = NEVER GIVE COMMAND.
\* 1 = GIVE COMMAND ON COLD BOOTS ONLY.
\* 2 = GIVE THE COMMAND ON WARM BOOTS ONLY.
\* 3 = GIVE THE COMMAND ON WARM AND COLD BOOTS.
\*\*\*\*\*

DEAB 01	AUTOFLG	DB 1	:AUTO COMMAND FEATURE
---------	---------	------	-----------------------

\*\*\*\*\*
\* IF THERE IS A COMMAND INSERTED HERE, IT WILL BE GIVEN IF THE
\* AUTO FEATURE IS ENABLED.
\* FOR EXAMPLE:
\*
\* COLDBEG DB 'MBASIC MYPROG'
\* COLDEND DB 0
\*
\* WILL EXECUTE MICROSOFT BASIC, AND MBASIC WILL EXECUTE THE
\* "MYPROG" BASIC PROGRAM.
\*\*\*\*\*

DEAC 5355424D49	COLDBEG	DB 'SUBMIT STARTUP';COLD BOOT COMMAND
DEBA 00	COLDEND	DB 0

DEBB 00      WARBEG DB      ''      ;WARM BOOT COMMAND GOES HERE  
 WARMEND DB      0

\*\*\*\*\*  
 \*  
 \* WBOOT LOADS IN ALL OF CP/M EXCEPT THE CBIOS, THEN INITIALIZES \*  
 \* SYSTEM PARAMETERS AS IN COLD BOOT. SEE THE COLD BOOT LOADER \*  
 \* LISTING FOR EXACTLY WHAT HAPPENS DURING WARM AND COLD BOOTS. \*  
 \*  
 \*\*\*\*\*

DEBC 310001      WBOOT      LXI      SP,TPA      ;SET UP STACK POINTER  
 DEBF 3E01      MVI      A,1  
 DEC0 =      WFLG      EQU      \$-1      ;TEST IF BEGINNING OR  
 DEC1 A7      ANA      A      ;      ENDING A WARM BOOT  
 DEC2 3E01      MVI      A,1  
 DEC4 32C0DE      STA      WFLG  
 DEC7 32AADE      STA      CWFLG      ;SET COLD/WARM BOOT FLAG  
 DECA CA58DE      JZ      GOCPM  
 DECD AF      XRA      A  
 DECE 32C0DE      STA      WFLG  
 DED1 4F      MOV      C,A  
  
 IF      (MAXHD NE 0) AND FIRST ;SUPPLY WARM BOOT FROM HARD DISK ?  
 LXI      H,CCP-200H      ;INITIAL DMA ADDRESS  
 PUSH      H  
 STA      HEAD  
 MVI      A,4  
 PUSH      PSW      ;SAVE FIRST SECTOR  
 CALL      HDDRV      ;SELECT DRIVE A  
 MVI      C,0  
 CALL      HDTRK      ;HOME THE DRIVE  
 WARMLOD      POP      PSW      ;RESTORE SECTOR  
 POP      H      ;RESTORE DMA ADDRESS  
 INR      A  
 STA      HDSECTR  
 CPI      16      ;PAST BDOS ?  
 JZ      WBOOT      ;YES, ALL DONE  
 INR      H      ;UPDATE DMA ADDRESS  
 INR      H  
 SHLD      HDADD  
 PUSH      H  
 PUSH      PSW  
 WARMRD      LXI      B,RETRIES\*100H+0 ;RETRY COUNTER  
 WRMREAD      PUSH      B      ;SAVE THE RETRY COUNT  
 CALL      HDREAD      ;READ THE SECTOR  
 POP      B  
 JNC      WARMLOD      ;TEST FOR ERROR  
 DCR      B      ;UPDATE THE ERROR COUNT  
 JNZ      WRMREAD      ;KEEP TRYING IF NOT TO MANY ERRORS  
 HLT  
 ENDIF  
  
 IF      (MAXFLOP NE 0) AND NOT FIRST ;SUPPLY WARM BOOT FROM 2D ?  
 CALL      DJDRV      ;SELECT DRIVE A  
 MVI      C,0      ;SELECT SINGLE DENSITY  
 DED2 CD33DD  
 DED5 0E00

DED7 CD2DF4	CALL	DJDEN		
DEDA ØEØØ	MVI	C,Ø	;SELECT SIDE Ø	
DEDC CD3ØF4	CALL	DJSIDE		
DEF ØEØF	MVI	A,15	;INITIALIZE THE SECTOR TO READ	
DEE1 32FFDE	STA	NEWSEC		
DEE4 21ØØC6	LXI	H,CCP-1ØØH	;AND THE DMA ADDRESS	
DEE7 221EDF	SHLD	NEWDMA		
DEEA CDFEDE	CALL	WARMLOD	;READ IN CP/M	
DEED Ø1ØØCC	LXI	B,CCP+5ØØH	;LOAD ADDRESS FOR REST OF WARM BOOT	
DEFØ CD12F4	CALL	DJDMA		
DEF3 ØEØ8	MVI	C,8		
DEF5 CDØFF4	CALL	DJSEC		
DEF8 CD32DF	CALL	WARMRD		
DEFB C3Ø3CC	JMP	CCP+5Ø3H		
DEFE 3EØF	WARMLOD	MVI	A,15	;PREVIOUS SECTOR
DEFF =	NEWSEC	EQU	\$-1	
DFØØ 3C		INR	A	;UPDATE THE PREVIOUS SECTOR
DFØ1 3C		INR	A	
DFØ2 FE1B		CPI	27	;WAS IT THE LAST ?
DFØ4 DA16DF		JC	NOWRAP	
DFØ7 D6Ø9		SUI	9	;YES
DFØ9 FE13		CPI	19	
DFØB C8		RZ		
DFØC 2A1EDF		LHLD	NEWDMA	
DFØF 118ØFB		LXI	D,-48ØH	
DF12 19		DAD	D	
DF13 221EDF		SHLD	NEWDMA	
DF16 32FFDE	NOWRAP	STA	NEWSEC	;SAVE THE NEW SECTOR TO READ
DF19 4F		MOV	C,A	
DF1A CDØFF4		CALL	DJSEC	
DF1D 21ØØC6		LXI	H,CCP-1ØØH	;GET THE PREVIOUS DMA ADDRESS
DF1E =	NEWDMA	EQU	\$-2	
DF2Ø 11ØØØ1		LXI	D,1ØØH	;UPDATE THE DMA ADDRESS
DF23 19		DAD	D	
DF24 221EDF		SHLD	NEWDMA	;SAVE THE DMA ADDRESS
DF27 44		MOV	B,H	
DF28 4D		MOV	C,L	
DF29 CD12F4		CALL	DJDMA	;SET THE DMA ADDRESS
DF2C CD32DF		CALL	WARMRD	
DF2F C3FEDE		JMP	WARMLOD	
DF32 Ø1ØØØA	WARMRD	LXI	B,RETRIES*1ØØH+Ø;MAXIMUM # OF ERRORS	
DF35 C5	WRMREAD	PUSH	B	
DF36 CDØCF4		CALL	DJTRK	;SET THE TRACK
DF39 CD15F4		CALL	DJREAD	;READ THE SECTOR
DF3C C1		POP	B	
DF3D DØ		RNC		;CONTINUE IF SUCCESSFUL
DF3E Ø5		DCR	B	
DF3F C235DF		JNZ	WRMREAD	;KEEP TRYING
DF42 C32AF4		JMP	DJERR	
		ENDIF		

\*\*\*\*\*

\* \* \*

\* SETSEC JUST SAVES THE DESIRED SECTOR TO SEEK TO UNTIL AN \*

\* ACTUAL READ OR WRITE IS ATTEMPTED.

```
DF45 60      SETSEC  MOV      H,B
DF46 69      MOV      L,C
DF47 22FDE4  SHLD     CPMSEC
DF4A C9      DONOP    RET
```

```
*****  
*  
* SETDMA SAVES THE DMA ADDRESS FOR THE DATA TRANSFER.  
*  
*****
```

```
DF4B 60      SETDMA  MOV      H,B      ;HL <- BC
DF4C 69      MOV      L,C
DF4D 22EDE0  SHLD     CPMDMA    ;CP/M DMA ADDRESS
DF50 C9      RET
```

```
*****  
*  
* HOME IS TRANSLATED INTO A SEEK TO TRACK ZERO.  
*  
*****
```

```
DF51 0E00  HOME  MVI      C,0      ;TRACK TO SEEK TO
```

```
*****  
*  
* SETTRK SAVES THE TRACK # TO SEEK TO. NOTHING IS DONE AT THIS  
* POINT, EVERYTHING IS DEFERRED UNTIL A READ OR WRITE.  
*  
*****
```

```
DF53 79      SETTRK  MOV      A,C      ;A <- TRACK #
DF54 3200E5  STA      CPMTRK   ;CP/M TRACK #
DF57 C9      RET
```

```
*****  
*  
* SECTRAN TRANSLATES A LOGICAL SECTOR # INTO A PHYSICAL SECTOR  
* #.  
*  
*****
```

```
DF58 3AFFE4  SECTRAN IF      (MAXHD NE 0) AND (MAXFLOP NE 0) ;BOTH TYPES ?
                  LDA      CPMDRV   ;GET THE DRIVE NUMBER
```

```
IF      FIRST
CPI    MAXHD*LOGDSK ;OVER THE # OF HARD DISKS ?
JC     TRANHD
```

```
ELSE
CPI    MAXFLOP    ;OVER THE # OF FLOPPIES ?
JNC    TRANHD
ENDIF
```

```
DF5B FE02
DF5D D28FDF
```

```

        ENDIF

        IF      (MAXHD EQ 0) OR (MAXFLOP EQ 0) ;JUST ONE TYPE ?

        SECTRAN EQU  $

        IF      MAXFLOP NE 0      ;FLOPPY TRANSLATION
        DF60 03      TRANFP  INX   B
        DF61 D5      PUSH    D      ;SAVE TABLE ADDRESS
        DF62 C5      PUSH    B      ;SAVE SECTOR #
        DF63 CD71E0  CALL    GETDPB ;GET DPB ADDRESS INTO HL
        DF66 7E      MOV     A,M   ;GET # OF CP/M SECTORS/TRACK
        DF67 B7      ORA     A      ;CLEAR CARY
        DF68 1F      RAR     ;DIVIDE BY TWO
        DF69 91      SUB    C
        DF6A F5      PUSH   PSW   ;SAVE ADJUSTED SECTOR
        DF6B FA77DF  JM    SIDETWO
        DF6E F1      SIDEA  POP   PSW   ;DISCARD ADJUSTED SECTOR
        DF6F C1      POP    B      ;RESTORE SECTOR REQUESTED
        DF70 D1      POP    D      ;RESTOR ADDRESS OF XLT TABLE
        DF71 EB      SIDEONE XCHG ;HL <- & (TRANSLATION TABLE)
        DF72 09      DAD    B      ;BC = OFFSET INTO TABLE
        DF73 6E      MOV    L,M   ;HL <- PHYSICAL SECTOR
        DF74 2600  MVI    H,0
        DF76 C9      RET

        DF77 010F00  SIDETWO LXI   B,15  ;OFFSET TO SIDE BIT
        DF7A 09      DAD    B
        DF7B 7E      MOV    A,M
        DF7C E608  ANI    8      ;TEST FOR DOUBLE SIDED
        DF7E CA6EDF  JZ    SIDEA ;MEDIA IS ONLY SINGLE SIDED
        DF81 F1      POP   PSW   ;RETRIEVE ADJUSTED SECTOR
        DF82 C1      POP   B
        DF83 2F      CMA
        DF84 3C      INR   A      ;MAKE SECTOR REQUEST POSITIVE
        DF85 4F      MOV    C,A   ;MAKE NEW SECTOR THE REQUESTED SECTOR
        DF86 D1      POP   D
        DF87 CD71DF  CALL   SIDEONE
        DF8A 3E80  MVI    A,80H ;SIDE TWO BIT
        DF8C B4      ORA    H      ;      AND SECTOR
        DF8D 67      MOV    H,A
        DF8E C9      RET
        ENDIF

        IF      MAXHD NE 0      ;HARD DISK TRANSLATION ROUTINE
        DF8F 60      TRANHD MOV   H,B
        DF90 69      MOV   L,C
        DF91 23      INX   H
        DF92 C9      RET
        ENDIF

```

```

*****
*          *          *
*  SETDRV SELECTS THE NEXT DRIVE TO BE USED IN READ/WRITE          *
*  OPERATIONS. IF THE DRIVE HAS NEVER BEEN SELECTED BEFORE, A          *
*          *          *

```

\* PARAMETER TABLE IS CREATED WHICH CORRECTLY DESCRIBES THE  
 \* DISKETTE CURRENTLY IN THE DRIVE. DISKETTES CAN BE OF FOUR  
 \* DIFFERENT SECTOR SIZES:  
 \*     1) 128 BYTES SINGLE DENSITY.  
 \*     2) 256 BYTES DOUBLE DENSITY.  
 \*     3) 512 BYTES DOUBLE DENSITY.  
 \*     4) 1024 BYTES DOUBLE DENSITY.  
 \*

```

DF93 79      SETDRV  MOV   A,C          ;SAVE THE DRIVE #
DF94 32FFE4   STA   CPMDRV
DF97 FE05     CPI   MAXFLOP+(MAXHD*LOGDSK) ;CHECK FOR A VALID DRIVE #
DF99 D262E0   JNC   ZRET          ;ILLEGAL DRIVE #
DF9C 7B       MOV   A,E          ;TEST IF DRIVE EVER LOGGED IN BEFORE
DF9D E601     ANI   1
DF9F C249E0   JNZ   SETDRV1      ;BIT 0 OF E = 0 -> NEVER SELECTED BEFORE

DFA2 3AFFE4   IF    (MAXHD NE 0) AND (MAXFLOP NE 0) ;BOTH TYPES ?
               LDA   CPMDRV      ;GET THE DRIVE NUMBER

               IF    FIRST
               CPI   MAXHD*LOGDSK ;OVER THE # OF HARD DISKS ?
               JC    DRVHD
               SUI   MAXHD*LOGDSK

DFA5 FE02     CPI   MAXFLOP      ;OVER THE # OF FLOPPIES ?
DFA7 D2FFDF   JNC   SUBFP
               ENDIF
               ENDIF

               IF    (MAXFLOP NE 0) AND FIRST
               MOV   C,A          ;SAVE DRIVE #
               MVI   A,0          ;HAVE THE FLOPPIES BEEN ACCESSED YET ?
FLOPFLG      EQU   $-1
               ANA   A
               JNZ   FLOPOK
               MVI   B,17         ;FLOPPIES HAVN'T BEEN ACCESSED
               LXI   H,DJBOOT     ;CHECK IF 2D CONTROLLER IS INSTALLED
               MVI   A,(JMP)
               CMP   M
               JNZ   ZRET
               DCR   B
               JNZ   CLOPP
               CALL  DJBOOT      ;INITIALIZE THE CONTROLLER
               MVI   A,1          ;SAVE 2D INITIALIZED FLAG
               STA   FLOPFLG
               ENDIF

               IF    MAXFLOP NE 0
DFAA 210100   FLOPOK LXI   H,1          ;SELECT SECTOR 1 OF TRACK 1
DFAD 2201E5   SHLD   TRUESEC
DFB0 3E01     MVI   A,1
DFB2 3200E5   STA   CPMTRK
DFB5 CD9EE1   CALL   FILL          ;FLUSH BUFFER AND REFILL
DFB8 DA62E0   JC    ZRET          ;TEST FOR ERROR RETURN
DFBB CD27F4   CALL   DJSTAT      ;GET STATUS ON CURRENT DRIVE

```

DFBE E60C	ANI	0CH	;STRIP OFF UNWANTED BITS
DFC0 F5	PUSH	PSW	;USED TO SELECT A DPB
DFC1 1F	RAR		
DFC2 218AE0	LXI	H,XLTS	;TABLE OF XLT ADDRESSES
DFC5 5F	MOV	E,A	
DFC6 1600	MVI	D,0	
DFC8 19	DAD	D	
DFC9 E5	PUSH	H	;SAVE POINTER TO PROPER XLT
DFCA CD71E0	CALL	GETDPB	;GET DPH POINTER INTO DE
DFCD EB	XCHG		;
DFCE D1	POP	D	
DFCF 0602	MVI	B,2	;NUMBER OF BYTES TO MOVE
DFD1 CDD4E1	CALL	MOVLOP	;MOVE THE ADDRESS OF XLT
DFD4 110800	LXI	D,8	;OFFSET TO DPB POINTER
DFD7 19	DAD	D	;HL <- &DPH.DPB
DFD8 E5	PUSH	H	
DFD9 2A07F0	LHLD	ORIGIN+7	;GET ADDRESS OF DJ TERMINAL OUT ROUTINE
DFDC 23	INX	H	;BUMP TO LOOK AT ADDRESS OF
			;UART STATUS LOCATION
DFDD 7E	MOV	A,M	
DFDE EE03	XRI	3	;ADJUST FOR PROPER REV DJ
DFE0 6F	MOV	L,A	
DFE1 26F3	MVI	H,(ORIGIN+300H)/100H	
DFE3 7E	MOV	A,M	
DFE4 E608	ANI	DBLSID	;CHECK DOUBLE SIDED BIT
DFE6 11FDE3	LXI	D,DPB128S	;BASE FOR SINGLE SIDED DPB'S
DFE9 C2EFDF	JNZ	SIDEOK	
DFEC 113DE4	LXI	D,DPB128D	;BASE OF DOUBLE SIDED DPB'S
DFEF EB	SIDEOK	XCHG	;HL <- DPB BASE, DE <- &DPH.DPB
dff0 D1	POP	D	;RESTORE DE (POINTER INTO DPH)
dff1 F1	POP	PSW	;OFFSET TO CORRECT DPB
dff2 17	RAL		
dff3 17	RAL		
dff4 4F	MOV	C,A	
dff5 0600	MVI	B,0	
dff7 09	DAD	B	
dff8 EB	XCHG		;PUT DPB ADDRESS IN DPH
dff9 73	MOV	M,E	
ffa 23	INX	H	
ffb 72	MOV	M,D	
	ENDIF		
DFFC C349E0	IF	(MAXHD NE 0) AND (MAXFLOP NE 0)	
	JMP	SETDRV1	;SKIP OVER THE HARD DISK SELECT
DFFF D602	SUBFP	IF	NOT FIRST
	SUI	MAXFLOP	;ADJUST THE DRIVE #
	ENDIF		
	ENDIF		
E001 CD68E0	DRVHD	IF	MAXHD NE 0
E004 79	CALL	DIVLOG	;DIVIDE BY LOGICAL DISKS PER DRIVE
E005 322AE3	MOV	A,C	
E008 CD18E3	STA	HDDISK	
E00B 7E	CALL	DRVPTR	
E00C 3C	MOV	A,M	
	INR	A	

CP/M MACRO ASSEM 2.0 #020 \*\*\* Cbios For CP/M Ver. 2.2 \*\*\*

E00D C249E0	JNZ	SETDRV1		
E010 F6FC	ORI	NULL	;SELECT DRIVE	
E012 D352	OUT	HDFUNC		
E014 3E05	MVI	A, SCENBL	;ENABLE THE CONTROLLER	
E016 D350	OUT	HDCNTL		
E018 0EEF	MVI	C, 239	;WAIT APPROX 2 MINUTES FOR DISK TO READY	
E01A 210000	LXI	H, 0		
E01D 2B	DCX	H		
E01E 7C	MOV	A, H		
E01F B5	ORA	L		
E020 CC66E0	CZ	DCRC		
E023 C8	RZ			
E024 DB50	IN	HDSTAT	;TEST IF READY YET	
E026 E620	ANI	DRVRDY		
E028 C21DE0	JNZ	TDELAY		
	IF	SDELAY		
E02B 210000	LXI	H, 0	;TIME ONE REVOLUTION OF THE DRIVE	
E02E 0E40	MVI	C, INDEX		
E030 DB50	IN	HDSTAT		
E032 A1	ANA	C		
E033 47	MOV	B, A	;SAVE CURRENT INDEX LEVEL IN B	
E034 DB50	INDX1	IN	HDSTAT	
E036 A1	ANA	C		
E037 B8	CMP	B	;LOOP UNTIL INDEX LEVEL CHANGES	
E038 CA34E0	JZ	INDX1		
E03B 23	INDX2	INX	H	
E03C DB50	IN	HDSTAT	;START COUNTING UNTIL INDEX RETURNS TO	
E03E A1	ANA	C	;PREVIOUS STATE	
E03F B8	CMP	B		
E040 C23BE0	JNZ	INDX2		
	IF	M10		
	DAD	H		
	ENDIF			
E043 2210E2	SHLD	SETTLE	;SAVE THE COUNT FOR TIMEOUT DELAY	
E046 CDFAE1	ENDIF	CALL	HDHOME	
		ENDIF		
E049 CD71E0	SETDRV1	CALL	GETDPB	;GET ADDRESS OF DPB IN HL
E04C 010F00		LXI	B, 15	;OFFSET TO SECTOR SIZE
E04F 09		DAD	B	
E050 7E		MOV	A, M	;GET SECTOR SIZE
E051 E607		ANI	7H	
E053 329EE0		STA	SECSIZ	
E056 7E		MOV	A, M	
E057 1F		RAR		
E058 1F		RAR		
E059 1F		RAR		
E05A 1F		RAR		
E05B E60F		ANI	0FH	
E05D 32DCE0		STA	SECPSEC	
E060 EB		XCHG		;HL <- DPH
E061 C9		RET		
E062 210000	ZRET	LXI	H, 0	;SELDRV ERROR EXIT

E065 C9 RET

E066 0D	DCRC	IF DCR	MAXHD NE 0
E067 C9		RET	C ;CONDITIONAL DECREMENT C ROUTINE

  

E068 0E00	DIVLOG	MVI C,0	
E06A D603	DIVLOGX	SUI LOGDSK	
E06C D8		RC	
E06D 0C		INR C	
E06E C36AE0		JMP DIVLOGX	
		ENDIF	

\*\*\*\*\*  
\* \*  
\* GETDPB RETURNS HL POINTING TO THE DPB OF THE CURRENTLY \*  
\* SELECTED DRIVE, DE POINTING TO DPH. \*  
\* \*  
\*\*\*\*\*

E071 3AFFE4	GETDPB	LDA CPMDRV	
E074 6F		MOV L,A	;FORM OFFSET
E075 2600		MVI H,0	
E077 29		DAD H	
E078 29		DAD H	
E079 29		DAD H	
E07A 29		DAD H	
E07B 11ADE4		LXI D,DPBASE	;BASE OF DPH'S
E07E 19		DAD D	
E07F E5		PUSH H	;SAVE ADDRESS OF DPH
E080 110A00		LXI D,10	;OFFSET TO DPB
E083 19		DAD D	
E084 7E		MOV A,M	;GET LOW BYTE OF DPB ADDRESS
E085 23		INX H	
E086 66		MOV H,M	;GET LOW BYTE OF DPB
E087 6F		MOV L,A	
E088 D1		POP D	
E089 C9		RET	

\*\*\*\*\*  
\* \*  
\* XLTS IS A TABLE OF ADDRESS THAT POINT TO EACH OF THE XLT \*  
\* TABLES FOR EACH SECTOR SIZE. \*  
\* \*  
\*\*\*\*\*

E08A 2FE3	XLTS	IF DW	MAXFLOP NE 0
E08C 4AE3		XLT128	;XLT FOR 128 BYTE SECTORS
E08E 7FE3		DW XLT256	;XLT FOR 256 BYTE SECTORS
E090 BCE3		DW XLT512	;XLT FOR 512 BYTE SECTORS
		DW XLT124	;XLT FOR 1024 BYTE SECTORS
		ENDIF	

\*\*\*\*\*  
\* \*  
\* WRITE ROUTINE MOVES DATA FROM MEMORY INTO THE BUFFER. IF THE \*  
\*\*\*\*\*

\* DESIRED CP/M SECTOR IS NOT CONTAINED IN THE DISK BUFFER, THE \*  
 \* BUFFER IS FIRST FLUSHED TO THE DISK IF IT HAS EVER BEEN \*  
 \* WRITTEN INTO, THEN A READ IS PERFORMED INTO THE BUFFER TO GET \*  
 \* THE DESIRED SECTOR. ONCE THE CORRECT SECTOR IS IN MEMORY, THE \*  
 \* BUFFER WRITTEN INDICATOR IS SET, SO THE BUFFER WILL BE \*  
 \* FLUSHED, THEN THE DATA IS TRANSFERRED INTO THE BUFFER. \*  
 \*

\*\*\*\*\*

E092 79	WRITE	MOV	A,C	;SAVE WRITE COMMAND TYPE
E093 3204E1		STA	WRITTYP	
E096 3E01		MVI	A,1	;SET WRITE COMMAND
E098 06		DB	(MVI) OR (B*8)	;THIS "MVI B" INSTRUCTION CAUSES ; ; THE FOLLOWING "XRA A" TO ; BE SKIPPED OVER.

\*\*\*\*\*

\*  
 \* READ ROUTINE TO BUFFER DATA FROM THE DISK. IF THE SECTOR \*  
 \* REQUESTED FROM CP/M IS IN THE BUFFER, THEN THE DATA IS SIMPLY \*  
 \* TRANSFERRED FROM THE BUFFER TO THE DESIRED DMA ADDRESS. IF \*  
 \* THE BUFFER DOES NOT CONTAIN THE DESIRED SECTOR, THE BUFFER IS \*  
 \* FLUSHED TO THE DISK IF IT HAS EVER BEEN WRITTEN INTO, THEN \*  
 \* FILLED WITH THE SECTOR FROM THE DISK THAT CONTAINS THE \*  
 \* DESIRED CP/M SECTOR.  
 \*

\*\*\*\*\*

E099 AF	READ	XRA	A	;SET THE COMMAND TYPE TO READ
E09A 32F0E0		STA	RDWR	;SAVE COMMAND TYPE

\*\*\*\*\*  
 \*  
 \* REDWRT CALCULATES THE PHYSICAL SECTOR ON THE DISK THAT \*  
 \* CONTAINS THE DESIRED CP/M SECTOR, THEN CHECKS IF IT IS THE \*  
 \* SECTOR CURRENTLY IN THE BUFFER. IF NO MATCH IS MADE, THE \*  
 \* BUFFER IS FLUSHED IF NECESSARY AND THE CORRECT SECTOR READ \*  
 \* FROM THE DISK.  
 \*

\*\*\*\*\*

E09D 0600	REDWRT	MVI	B,0	;THE 0 IS MODIFIED TO CONTAIN THE LOG2
E09E =		SECSIZ	EQU	\$-1 ; OF THE PHYSICAL SECTOR SIZE/128 ; ON THE CURRENTLY SELECTED DISK.
E09F 2AFDE4		LHLD	CPMSEC	;GET THE DESIRED CP/M SECTOR #
E0A2 7C		MOV	A,H	
E0A3 E680		ANI	80H	;SAVE ONLY THE SIDE BIT
E0A5 4F		MOV	C,A	;REMEMBER THE SIDE
E0A6 7C		MOV	A,H	
E0A7 E67F		ANI	7FH	;FORGET THE SIDE BIT
E0A9 67		MOV	H,A	
E0AA 2B		DCX	H	;TEMPORARY ADJUSTMENT
E0AB 05	DIVLOOP	DCR	B	;UPDATE REPEAT COUNT
E0AC CAB9E0		JZ	DIVDONE	
E0AF B7		ORA	A	
E0B0 7C		MOV	A,H	

E0B1 1F	RAR			
E0B2 67	MOV	H, A		
E0B3 7D	MOV	A, L		
E0B4 1F	RAR			; <div style="text-align: center;">;DIVIDE THE CP/M SECTOR # BY THE SIZE ;OF THE PHYSICAL SECTORS</div>
E0B5 6F	MOV	L, A		
E0B6 C3ABE0	JMP	DIVLOOP		;
E0B9 23	DIVDONE	INX	H	
E0BA 7C		MOV	A, H	
E0BB B1		ORA	C	; <div style="text-align: center;">RESTORE THE SIDE BIT</div>
E0BC 67		MOV	H, A	
E0BD 2201E5	SHLD	TRUESEC		; <div style="text-align: center;">SAVE THE PHYSICAL SECTOR NUMBER</div>
E0C0 21FFE4	LXI	H, CPMDRV		; <div style="text-align: center;">POINTER TO DESIRED DRIVE, TRACK, AND SECTOR</div>
E0C3 1103E5	LXI	D, BUFDRV		; <div style="text-align: center;">POINTER TO BUFFER DRIVE, TRACK, AND SECTOR</div>
E0C6 0605	MVI	B, 5		; <div style="text-align: center;">COUNT LOOP</div>
E0C8 05	DTSLOP	DCR	B	; <div style="text-align: center;">TEST IF DONE WITH COMPARE</div>
E0C9 CAD7E0		JZ	MOVE	; <div style="text-align: center;">YES, MATCH. GO MOVE THE DATA</div>
E0CC 1A		LDAX	D	; <div style="text-align: center;">GET A BYTE TO COMPARE</div>
E0CD BE		CMP	M	; <div style="text-align: center;">TEST FOR MATCH</div>
E0CE 23		INX	H	; <div style="text-align: center;">BUMP POINTERS TO NEXT DATA ITEM</div>
E0CF 13		INX	D	
E0D0 CAC8E0		JZ	DTSLOP	; <div style="text-align: center;">MATCH, CONTINUE TESTING</div>
 ***** * * DRIVE, TRACK, AND SECTOR DON'T MATCH, FLUSH THE BUFFER IF * NECESSARY AND THEN REFILL. * *****				
E0D3 CD9EE1	CALL	FILL		; <div style="text-align: center;">FILL THE BUFFER WITH CORRECT PHYSICAL SECTOR</div>
E0D6 D8	RC			; <div style="text-align: center;">NO GOOD, RETURN WITH ERROR INDICATION</div>
 ***** * * MOVE HAS BEEN MODIFIED TO CAUSE EITHER A TRANSFER INTO OR OUT * THE BUFFER. * *****				
E0D7 3AFDE4	MOVE	LDA	CPMSEC	; <div style="text-align: center;">GET THE CP/M SECTOR TO TRANSFER</div>
E0DA 3D		DCR	A	; <div style="text-align: center;">ADJUST TO PROPER SECTOR IN BUFFER</div>
E0DB E600		ANI	0	; <div style="text-align: center;">STRIP OFF HIGH ORDERED BITS</div>
E0DC =	SECPSEC	EQU	\$-1	; <div style="text-align: center;">THE 0 IS MODIFIED TO REPRESENT THE # OF CP/M SECTORS PER PHYSICAL SECTORS</div>
E0DD 6F	MOV	L, A		; <div style="text-align: center;">PUT INTO HL</div>
E0DE 2600	MVI	H, 0		; <div style="text-align: center;">FORM OFFSET INTO BUFFER</div>
E0E0 29	DAD	H		
E0E1 29	DAD	H		
E0E2 29	DAD	H		
E0E3 29	DAD	H		
E0E4 29	DAD	H		
E0E5 29	DAD	H		
E0E6 29	DAD	H		
E0E7 1107E5	LXI	D, BUFFER		; <div style="text-align: center;">BEGINNING ADDRESS OF BUFFER</div>
E0EA 19	DAD	D		; <div style="text-align: center;">FORM BEGINNING ADDRESS OF SECTOR TO TRANSFER</div>

E0EB EB	XCHG			
E0EC 210000	LXI	H,0		;DE = ADDRESS IN BUFFER
				;GET DMA ADDRESS, THE 0 IS MODIFIED TO
				;CONTAIN THE DMA ADDRESS
E0ED =	CPMDMA	EQU	\$-2	
E0EF 3E00		MVI	A,0	;THE ZERO GETS MODIFIED TO CONTAIN
				A ZERO IF A READ, OR A 1 IF WRITE
E0F0 =	RDWR	EQU	\$-1	
E0F1 A7		ANA	A	;TEST WHICH KIND OF OPERATION
E0F2 C2FAE0		JNZ	INTO	;TRANSFER DATA INTO THE BUFFER
E0F5 CDD2E1	OUTOF	CALL	MOVER	
E0F8 AF		XRA	A	
E0F9 C9		RET		
E0FA EB	INTO	XCHG		
E0FB CDD2E1		CALL	MOVER	;MOVE THE DATA, HL = DESTINATION
				;DE = SOURCE
E0FE 3E01		MVI	A,1	
E100 320DE1		STA	BUFWRTN	;SET BUFFER WRITTEN INTO FLAG
E103 3E00		MVI	A,0	;CHECK FOR DIRECTORY WRITE
E104 =	WRITTYP	EQU	\$-1	
E105 3D		DCR	A	
E106 3E00		MVI	A,0	
E108 3204E1		STA	WRITTYP	;SET NO DIRECTORY WRITE
E10B C0		RNZ		;NO ERROR EXIT

\*\*\*\*\*
\* \* FLUSH WRITES THE CONTENTS OF THE BUFFER OUT TO THE DISK IF \*
\* IT HAS EVER BEEN WRITTEN INTO. \*
\*\*\*\*\*

E10C 3E00	FLUSH	MVI	A,0	;THE 0 IS MODIFIED TO REFLECT IF
				;THE BUFFER HAS BEEN WRITTEN INTO

E10D =	BUFWRTN	EQU	\$-1	
E10E A7		ANA	A	;TEST IF WRITTEN INTO
E10F C8		RZ		;NOT WRITTEN, ALL DONE

E110 2118F4	IF	(MAXHD NE 0) AND (MAXFLOP NE 0)		
E113 119AE2	LXI	H,DJWRITE		;WRITE OPERATION FOR DISK JOCKEY
E116 CDE1E1	LXI	D,HDWRITE		;WRITE OPERATION FOR HARD DISK
	CALL	DECIDE		
	ELSE			
	IF	MAXHD NE 0		
	LXI	H,HDWRITE		
	ENDIF			
	IF	MAXFLOP NE 0		
	LXI	H,DJWRITE		
	ENDIF			
	ENDIF			

\*\*\*\*\*
\* \* PREP PREPARES TO READ/WRITE THE DISK. RETRIES ARE ATTEMPTED. \*
\* UPON ENTRY, H&L MUST CONTAIN THE READ OR WRITE OPERATION \*
\* ADDRESS. \*

\*  
\*\*\*\*\*  
E119 AF PREP XRA A ;RESET BUFFER WRITTEN FLAG  
E11A 320DE1 STA BUFWRTN  
E11D 227FE1 SHLD RETRYOP ;SET UP THE READ/WRITE OPERATION  
E120 060A MVI B, RETRIES ;MAXIMUM NUMBER OF RETRIES TO ATTEMPT  
E122 C5 RETRYLP PUSH B ;SAVE THE RETRY COUNT  
E123 3A03E5 LDA BUFDRV ;GET DRIVE NUMBER INVOLVED IN THE OPERATION  
  
IF (MAXHD NE 0) AND (MAXFLOP NE 0)  
IF FIRST  
CPI MAXHD\*LOGDSK  
JC NOADJST  
SUI MAXHD\*LOGDSK  
ELSE  
CPI MAXFLOP  
JC NOADJST  
SUI MAXFLOP  
ENDIF  
  
E126 FE02  
E128 DA2DE1  
E12B D602  
  
E12D 4F NOADJST MOV C,A  
E12E 2133DD LXI H,DJDRV ;SELECT DRIVE  
E131 11E9E1 LXI D,HDDRV  
E134 CDDDE1 CALL DECIDGO  
ELSE  
MOV C,A  
IF MAXHD NE 0  
CALL HDDRV  
ENDIF  
IF MAXFLOP NE 0  
CALL DJDRV ;SELECT THE DRIVE  
ENDIF  
ENDIF  
  
E137 3A04E5 LDA BUFTRK  
E13A A7 ANA A ;TEST FOR TRACK ZERO  
E13B 4F MOV C,A  
E13C C5 PUSH B  
  
IF (MAXHD NE 0) AND (MAXFLOP NE 0)  
LXI H,DJHOME  
LXI D,HDHOME  
CZ DECIDGO  
ELSE  
IF MAXHD NE 0  
CZ HDHOME  
ENDIF  
IF MAXFLOP NE 0  
CZ DJHOME ;HOME THE DRIVE IF TRACK 0  
ENDIF  
ENDIF  
  
E146 C1 POP B ;RESTORE TRACK #  
IF (MAXHD NE 0) AND (MAXFLOP NE 0)

```

E147 210CF4      LXI    H,DJTRK
E14A 111BE2      LXI    D,HDTRK
E14D CDDDE1      CALL   DECIDGO
ELSE
IF    MAXHD NE 0
CALL  HDTRK
ENDIF
IF    MAXFLOP NE 0
CALL  DJTRK      ;SEEK TO PROPER TRACK
ENDIF
ENDIF

E150 2A05E5      LHLD   BUFSEC
E153 7C          MOV    A,H      ;GET SECTOR INVOLVED IN OPERATION
E154 07          RLC    0          ;BIT 0 OF A EQUALS SIDE #
E155 E601          ANI    1          ;STRIP OFF UNNECESSARY BITS
E157 4F          MOV    C,A      ;C <- SIDE #

IF    (MAXHD NE 0) AND (MAXFLOP NE 0)
LXI  H,DJSIDE
LXI  D,HDSIDE
CALL DECIDGO
ELSE
IF    MAXHD NE 0
CALL HDSIDE
ENDIF
IF    MAXFLOP NE 0
CALL DJSIDE      ;SELECT THE SIDE
ENDIF
ENDIF

E161 2A05E5      LHLD   BUFSEC
E164 7C          MOV    A,H
E165 E67F          ANI    7FH      ;STRIP OFF SIDE BIT
E167 47          MOV    B,A      ;C <- SECTOR #
E168 4D          MOV    C,L

IF    (MAXHD NE 0) AND (MAXFLOP NE 0)
LXI  H,DJSEC
LXI  D,HDSEC
CALL DECIDGO
ELSE
IF    MAXHD NE 0
CALL HDSEC
ENDIF
IF    MAXFLOP NE 0
CALL DJSEC      ;SELECT THE SIDE
ENDIF
ENDIF

E172 0107E5      LXI    B,BUFFER ;SET THE DMA ADDRESS
IF    (MAXHD NE 0) AND (MAXFLOP NE 0)
LXI  H,DJDMA
LXI  D,HDDMA
CALL DECIDGO

```

```

ELSE
IF      MAXHD NE 0
CALL    HDDMA
ENDIF
IF      MAXFLOP NE 0
CALL   DJDMA      ;SELECT THE SIDE
ENDIF
ENDIF

```

```

E17E CD0000  CALL  0          ;GET OPERATION ADDRESS
E17F =      RETRYOP EQU   $-2
E181 C1      POP   B          ;RESTORE THE RETRY COUNTER
E182 3E00    MVI   A,0        ;NO ERROR EXIT STATUS
E184 D0      RNC   B          ;RETURN NO ERROR
E185 05      DCR   B          ;UPDATE THE RETRY COUNTER
E186 37      STC   B          ;ASSUME RETRY COUNT EXPIRED
E187 3EFF    MVI   A,0FFH    ;ERROR RETURN
E189 C8      RZ
E18A 78      MOV   A,B
E18B FE05    CPI   RETRIES/2
E18D C222E1  JNZ   RETRYLP  ;TRY AGAIN

```

```

E190 C5      PUSH  B
IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
LXI   H,DJHOME
LXI   D,HDHOME
CALL  DECIDGO
ELSE
IF      MAXHD NE 0
CALL  HDHOME
ENDIF
IF      MAXFLOP NE 0
CALL  DJHOME      ;HOME THE DRIVE IF TRACK 0
ENDIF
ENDIF

```

```

E19A C1      POP   B
E19B C322E1  JMP   RETRYLP

```

```

*****
*          *
* FILL FILLS THE BUFFER WITH A NEW SECTOR FROM THE DISK. *
*          *
*****
```

```

E19E CD0CE1  FILL  CALL  FLUSH      ;FLUSH BUFFER FIRST
E1A1 D8      FILL  RC      ;CHECK FOR ERROR
E1A2 11FFE4  LXI   D,CPMDRV ;UPDATE THE DRIVE, TRACK, AND SECTOR
E1A5 2103E5  LXI   H,BUFDRV
E1A8 0604    MVI   B,4      ;NUMBER OF BYTES TO MOVE
E1AA CDD4E1  CALL  MOVLOP    ;COPY THE DATA
E1AD 3AF0E0  LDA   RDWR
E1B0 A7      ANA   A
E1B1 CAC6E1  JZ    FREAD
E1B4 3A04E1  LDA   WRITYP

```

```

E1B7 3D          DCR    A
E1B8 3D          DCR    A
E1B9 C8          RZ
E1BA CD71E0      CALL   GETDPB
E1BD 110F00      LXI    D,15
E1C0 19          DAD    D
E1C1 7E          MOV    A,M
E1C2 E603        ANI    3
E1C4 3D          DCR    A
E1C5 C8          RZ

E1C6 =          FREAD EQU   $
IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E1C6 2115F4      LXI    H,DJREAD
E1C9 1165E2      LXI    D,HDREAD
E1CC CDE1E1      CALL   DECIDE
ELSE
IF      MAXHD NE 0
LXI    H,HDREAD
ENDIF
IF      MAXFLOP NE 0
LXI   H,DJREAD      ;SELECT THE SIDE
ENDIF
ENDIF
ENDIF
E1CF C319E1      JMP    PREP      ;SELECT DRIVE, TRACK, AND SECTOR.
                                ; THEN READ THE BUFFER
*****
* * MOVER MOVES 128 BYTES OF DATA. SOURCE POINTER IN DE, DEST
* * POINTER IN HL.
* *
*****
E1D2 0680        MOVER  MVI    B,128      ;LENGTH OF TRANSFER
E1D4 1A          MOVLOP LDAX   D          ;GET A BTE OF SOURCE
E1D5 77          MOV    M,A          ;MOVE IT
E1D6 13          INX    D          ;BUMP POINTERS
E1D7 23          INX    H
E1D8 05          DCR    B          ;UPDATE COUNTER
E1D9 C2D4E1      JNZ    MOVLOP      ;CONTINUE MOVING UNTIL DONE
E1DC C9          RET

*****
* * ROUTINES TO DECIDE WHICH CONTROLLER TO USE.
* *
*****
E1DD CDE1E1      DECIDGO IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E1E0 E9          DECIDGO CALL   DECIDE ;WHICH CONTROLLER ?
                                PCHL
                                ENDIF

E1E1 3A03E5      DECIDE IF      (MAXHD NE 0) AND (MAXFLOP NE 0)
E1E1 3A03E5      DECIDE LDA    BUFDRV      ;GET PROPER ROUTINE INTO H&L, BASED

```

```

IF      FIRST      ;  ON CURRENTLY SELECTED DRIVE
CPI    MAXHD*LOGDSK
RNC
ELSE
E1E4 FE02
E1E6 D8
CPI    MAXFLOP
RC
ENDIF
E1E7 EB
XCHG
RET
ENDIF

```

```

*****
* THE FOLLOWING IS THE EQUIVALENT OF THE LOWEST LEVEL DRIVERS
* FOR THE HARD DISK.
*****

```

E1E9 79	HDDRV	IF	MAXHD NE 0	
E1EA CD68E0		MOV	A,C	;SELECT HARD DISK DRIVE
E1ED 79		CALL	DIVLOG	;GET THE PHYSICAL DRIVE #
E1EE 322AE3		MOV	A,C	
E1F1 F6FC		STA	HDDISK	;SELECT THE DRIVE
E1F3 D352		ORI	NULL	
E1F5 3E0F		OUT	HDFUNC	
E1F7 D350		MVI	A,WENABL	
E1F9 C9		OUT	HDCNTL	
	RET			
E1FA CD18E3	HDHOME	CALL	DRVPTR	
E1FD 3600		MVI	M,0	;SET TRACK TO ZERO
		IF	SDELAY	
E1FF DB50		IN	HDSTAT	;TEST STATUS
E201 E601		ANI	TKZERO	;AT TRACK ZERO ?
E203 CA0FE2		JZ	DELAY	
E206 3E01		MVI	A,1	
E208 37		STC		
E209 CD2FE2		CALL	ACCOK	;TAKE ONE STEP OUT
E20C C3FFE1	JMP	STEPO		
	ELSE			
	IN	HDSTAT		
	ANI	TKZERO		
	RZ			
	XRA	A		
	JMP	ACCOK		
	ENDIF			
E20F 210000	DELAY	IF	SDELAY	
E210 =		LXI	H,0	;GET DELAY
E212 2B		EQU	\$-2	
E213 7C		DCX	H	;WAIT 20MS
E214 B5		MOV	A,H	
	ORA	L		

E215 23		INX	H	
E216 2B		DCX	H	
E217 C212E2		JNZ	DELOOP	
E21A C9		RET		
		ENDIF		
E21B CD18E3	HDTRK	CALL	DRVPTR	;GET POINTER TO CURRENT TRACK
E21E 5E		MOV	E, M	;GET CURRENT TRACK
E21F 71		MOV	M, C	;UPDATE THE TRACK
E220 7B		MOV	A, E	;NEED TO SEEK AT ALL ?
E221 91		SUB	C	
E222 C8		RZ		
E223 3F		CMC		;GET CARRY INTO DIRECTION
E224 DA29E2		JC	HDTRK2	
E227 2F		CMA		
E228 3C		INR	A	
		IF	NOT SDELAY	
	HDTRK2	JMP	ACCOCK	
		ELSE		
E229 CD2FE2	HDTRK2	CALL	ACCOCK	
E22C C30FE2		JMP	DELAY	
		ENDIF		
E22F 47	ACCOCK	MOV	B, A	;PREP FOR BUILD
E230 CD23E3		CALL	BUILD	
E233 E6FB	SLOOP	ANI	NSTEP	;GET STEP PULSE LOW
E235 D352		OUT	HDFUNC	;OUTPUT LOW STEP LINE
E237 F604		ORI	PSTEP	;SET STEP LINE HIGH
E239 D352		OUT	HDFUNC	;OUTPUT HIGH STEP LINE
E23B 05		DCR	B	;UPDATE REPEAT COUNT
E23C C233E2		JNZ	SLOOP	;KEEP GOING THE REQUIRED # OF TRACKS
E23F C348E2		JMP	WSDONE	
E242 60	HDDMA	MOV	H, B	;SAVE THE DMA ADDRESS
E243 69		MOV	L, C	
E244 227FE2		SHLD	HDADD	
E247 =	HDSIDE	EQU	\$	
E247 C9		RET		
E248 DB50	WSDONE	IN	HDSTAT	
E24A E604		ANI	COMPLT	
E24C CA48E2		JZ	WSDONE	
E24F C9		RET		
E250 3E1F	HDSEC	IF	M26	
E252 A1		MVI	A, 01FH	;FOR COMPATIBILITY WITH CBIOS REV 2.3, 2.4
E253 CC62E2		ANA	C	
E256 3208E3		CZ	GETSPT	
E259 3EE0		STA	HDSECTR	
E25B A1		MVI	A, 0E0H	
E25C 07		ANA	C	
E25D 07		RLC		
E25E 07		RLC		
E25F 3224E3		RLC		
E262 3E20	GETSPT	STA	HEAD	
		MVI	A, HDSPT	

E264 C9

RET

ELSE

HDSEC MOV A,C  
CALL DIVSPT  
ADI HDSPT  
ANA A  
CZ GETSPT  
STA HDSECTR  
MOV A,C  
STA HEAD  
GETSPT MVI A,HDSPT  
DCR C  
RET

DIVSPT MVI C,Ø  
DIVSPTX SUI HDSPT  
RC  
INR C  
JMP DIVSPTX  
ENDIF

E265 CDE3E2 HDREAD CALL HDPREP  
E268 D8 RC  
E269 AF XRA A  
E26A D351 OUT HDCMND  
E26C 2F CMA  
E26D D353 OUT HDDATA  
E26F D353 OUT HDDATA  
E271 3E01 MVI A,RSECT ;READ SECTOR COMMAND  
E273 D351 OUT HDCMND  
E275 CDC9E2 CALL PROCESS  
E278 D8 RC  
E279 AF XRA A  
E27A D351 OUT HDCMND  
E27C Ø680 MVI B,SECLEN/4  
E27E 210000 LXI H,Ø  
E27F = HDADD EQU \$-2  
E281 DB53 IN HDDATA  
E283 DB53 IN HDDATA  
E285 DB53 RTLOOP IN HDDATA ;MOVE FOUR BYTES  
E287 77 MOV M,A  
E288 23 INX H  
E289 DB53 IN HDDATA  
E28B 77 MOV M,A  
E28C 23 INX H  
E28D DB53 IN HDDATA  
E28F 77 MOV M,A  
E290 23 INX H  
E291 DB53 IN HDDATA  
E293 77 MOV M,A  
E294 23 INX H  
E295 Ø5 DCR B  
E296 C285E2 JNZ RTLOOP  
E299 C9 RET

E29A CDE3E2	HDWRITE	CALL	HDPREP	:PREPARE HEADER
E29D D8			RC	
E29E AF			XRA	A
E29F D351			OUT	HDCMND
E2A1 2A7FE2			LHLD	HDADD
E2A4 0680			MVI	B, SECLEN/4
E2A6 7E	WTLOOP		MOV	A,M ;MOVE 4 BYTES
E2A7 D353			OUT	HDDATA
E2A9 23			INX	H
E2AA 7E			MOV	A,M
E2AB D353			OUT	HDDATA
E2AD 23			INX	H
E2AE 7E			MOV	A,M
E2AF D353			OUT	HDDATA
E2B1 23			INX	H
E2B2 7E			MOV	A,M
E2B3 D353			OUT	HDDATA
E2B5 23			INX	H
E2B6 05			DCR	B
E2B7 C2A6E2			JNZ	WTLOOP
E2BA 3E05			MVI	A,WSECT ;ISSUE WRITE SECTOR COMMAND
E2BC D351			OUT	HDCMND
E2BE CDC9E2		CALL	PROCESS	
E2C1 D8			RC	
E2C2 3E10			MVI	A,WFAULT
E2C4 A0			ANA	B
E2C5 37			STC	
E2C6 C8			RZ	
E2C7 AF			XRA	A
E2C8 C9			RET	
E2C9 DB50	PROCESS	IN	HDSTAT	:WAIT FOR COMMAND TO FINISH
E2CB 47		MOV	B,A	
E2CC E602		ANI	OPDONE	
E2CE CAC9E2		JZ	PROCESS	
E2D1 3E07		MVI	A,DSKCLK	
E2D3 D350		OUT	HDCNTL	
E2D5 DB50		IN	HDSTAT	
E2D7 E608		ANI	TMOUT	:TIMED OUT ?
E2D9 37		STC		
E2DA C0		RNZ		
E2DB DB51		IN	HDRESLT	
E2DD E602		ANI	RETRY	:ANY RETRIES ?
E2DF 37		STC		
E2E0 C0		RNZ		
E2E1 AF		XRA	A	
E2E2 C9		RET		
E2E3 DB50	HDPREP	IN	HDSTAT	
E2E5 E620		ANI	DRVRDY	
E2E7 37		STC		
E2E8 C0		RNZ		
E2E9 3E08		MVI	A,ISBUFF	:INITIALIZE POINTER
E2EB D351		OUT	HDCMND	
E2ED CD23E3		CALL	BUILD	

E2F0 F60C		ORI	ØCH	
E2F2 D352		OUT	HDFUNC	
E2F4 3A24E3		LDA	HEAD	
E2F7 D353		OUT	HDDATA	; FORM HEAD BYTE
E2F9 CD18E3		CALL	DRVPTR	
E2FC 7E		MOV	A,M	; FORM TRACK BYTE
E2FD D353		OUT	HDDATA	
E2FF A7		ANA	A	
E300 Ø680		MVI	B,8ØH	
E302 CA07E3		JZ	ZKEY	
E305 Ø600		MVI	B,Ø	
E307 3E00	ZKEY	MVI	A,Ø	; FORM SECTOR BYTE
E308 =	HDSECTR	EQU	\$-1	
E309 D353		OUT	HDDATA	
E30B 78		MOV	A,B	
E30C D353		OUT	HDDATA	
E30E 3E07		MVI	A,DSKCLK	
E310 D350		OUT	HDCNTL	
E312 3E0F		MVI	A,WENABL	
E314 D350		OUT	HDCNTL	
E316 AF		XRA	A	
E317 C9		RET		
E318 2A2AE3	DRVPTR	LHLD	HDDISK	
E31B EB		XCHG		
E31C 1600		MVI	D,Ø	
E31E 212EE3		LXI	H,DRIVES	
E321 19		DAD	D	
E322 C9		RET		
E323 3E00	BUILD	MVI	A,Ø	
E324 =	HEAD	EQU	\$-1	
E325 17		RAL		
E326 17		RAL		
E327 17		RAL		
E328 17		RAL		
E329 F600		ORI	Ø	
E32A =	HDDISK	EQU	\$-1	
E32B EEFØ		XRI	ØFØH	
E32D C9		RET		
E32E =	DRIVES	EQU	\$	
		REPT	MAXHD	
		DB	ØFFH	
		ENDM		
E32E+FF		DB	ØFFH	
		ENDIF		

\*\*\*\*\*
\* XLT TABLES (SECTOR SKEW TABLES) FOR CP/M 2.0. THESE TABLES \*
\* DEFINE THE SECTOR TRANSLATION THAT OCCURS WHEN MAPPING CP/M \*
\* SECTORS TO PHYSICAL SECTORS ON THE DISK. THERE IS ONE SKEW \*
\* TABLE FOR EACH OF THE POSSIBLE SECTOR SIZES. CURRENTLY THE \*
\* TABLES ARE LOCATED ON TRACK 0 SECTORS 6 AND 8. THEY ARE \*
\* LOADED INTO MEMORY IN THE CBIOS RAM BY THE COLD BOOT ROUTINE. \*

\*  
\*\*\*\*\*

			IF	MAXFLOP NE Ø
E32F	00	XLT128	DB	Ø
E330	01070D1319		DB	1, 7, 13, 19, 25
E335	050B1117		DB	5, 11, 17, 23
E339	03090F15		DB	3, 9, 15, 21
E33D	02080E141A		DB	2, 8, 14, 20, 26
E342	060C1218		DB	6, 12, 18, 24
E346	040A1016		DB	4, 10, 16, 22
E34A	00	XLT256	DB	Ø
E34B	0102131425		DB	1, 2, 19, 20, 37, 38
E351	0304151627		DB	3, 4, 21, 22, 39, 40
E357	0506171829		DB	5, 6, 23, 24, 41, 42
E35D	0708191A2B		DB	7, 8, 25, 26, 43, 44
E363	090A1B1C2D		DB	9, 10, 27, 28, 45, 46
E369	0B0C1D1E2F		DB	11, 12, 29, 30, 47, 48
E36F	0D0E1F2031		DB	13, 14, 31, 32, 49, 50
E375	0F10212233		DB	15, 16, 33, 34, 51, 52
E37B	11122324		DB	17, 18, 35, 36
E37F	00	XLT512	DB	Ø
E380	0102030411		DB	1, 2, 3, 4, 17, 18, 19, 20
E388	2122232431		DB	33, 34, 35, 36, 49, 50, 51, 5
E390	0506070815		DB	5, 6, 7, 8, 21, 22, 23, 24
E398	2526272835		DB	37, 38, 39, 40, 53, 54, 55, 5
E3A0	090A0B0C19		DB	9, 10, 11, 12, 25, 26, 27, 28
E3A8	292A2B2C39		DB	41, 42, 43, 44, 57, 58, 59, 6
E3B0	0D0E0F101D		DB	13, 14, 15, 16, 29, 30, 31, 3
E3B8	2D2E2F30		DB	45, 46, 47, 48
E3BC	00	XLT124	DB	Ø
E3BD	0102030405		DB	1, 2, 3, 4, 5, 6, 7, 8
E3C5	191A1B1C1D		DB	25, 26, 27, 28, 29, 30, 31, 3
E3CD	3132333435		DB	49, 50, 51, 52, 53, 54, 55, 5
E3D5	090A0B0C0D		DB	9, 10, 11, 12, 13, 14, 15, 16
E3DD	2122232425		DB	33, 34, 35, 36, 37, 38, 39, 4
E3E5	393A3B3C3D		DB	57, 58, 59, 60, 61, 62, 63, 6
E3ED	1112131415		DB	17, 18, 19, 20, 21, 22, 23, 2
E3F5	292A2B2C2D		DB	41, 42, 43, 44, 45, 46, 47, 4

\*\*\*\*\*  
\*  
\* EACH OF THE FOLLOWING TABLES DESCRIBES A DISKETTE WITH THE  
\* SPECIFIED CHARACTERISTICS.  
\*

```
*****  
*  
* THE FOLLOWING DPB DEFINES A DISKETTE FOR 128 BYTE SECTORS,  
* SINGLE DENSITY, AND SINGLE SIDED.  
*
```

\*\*\*\*\*

E3FD 1A00	DPB128S	DW	26	;CP/M SECTORS/TRACK
E3FF 03		DB	3	;BSH
E400 07		DB	7	;BLM
E401 00		DB	0	;EXM
E402 F200		DW	242	;DSM
E404 3F00		DW	63	;DRM
E406 C0		DB	0C0H	;AL0
E407 00		DB	0	;AL1
E408 1000		DW	16	;CKS
E40A 0200		DW	2	;OFF
E40C 01		DB	1H	;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) + ;LOG2(#BYTES PER SECTOR/128) + 1 + ;8 IF DOUBLE SIDED.

\*\*\*\*\*

\* THE FOLLOWING DPB DEFINES A DISKETTE FOR 256 BYTE SECTORS,  
\* DOUBLE DENSITY, AND SINGLE SIDED.  
\*

E40D 3400	DPB256S	DW	52	;CP/M SECTORS/TRACK
E40F 04		DB	4	;BSH
E410 0F		DB	15	;BLM
E411 00		DB	0	;EXM
E412 F200		DW	242	;DSM
E414 7F00		DW	127	;DRM
E416 C0		DB	0C0H	;AL0
E417 00		DB	0	;AL1
E418 2000		DW	32	;CKS
E41A 0200		DW	2	;OFF
E41C 12		DB	12H	;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) + ;LOG2(#BYTES PER SECTOR/128) + 1 + ;8 IF DOUBLE SIDED.

\*\*\*\*\*

\* THE FOLLOWING DPB DEFINES A DISKETTE AS 512 BYTE SECTORS,  
\* DOUBLE DENSITY, AND SINGLE SIDED.  
\*

E41D 3C00	DPB512S	DW	60	;CP/M SECTORS/TRACK
E41F 04		DB	4	;BSH
E420 0F		DB	15	;BLM
E421 00		DB	0	;EXM
E422 1801		DW	280	;DSM
E424 7F00		DW	127	;DRM
E426 C0		DB	0C0H	;AL0
E427 00		DB	0	;AL1
E428 2000		DW	32	;CKS
E42A 0200		DW	2	;OFF
E42C 33		DB	33H	;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) + ;LOG2(#BYTES PER SECTOR/128) + 1 +

;8 IF DOUBLE SIDED.

```
*****
* THE FOLLOWING DPB DEFINES A DISKETTE AS 1024 BYTE SECTORS,
* DOUBLE DENSITY, AND SINGLE SIDED.
*****
```

E42D 4000	DP1024S	DW	64	;CP/M SECTORS/TRACK
E42F 04		DB	4	;BSH
E430 0F		DB	15	;BLM
E431 00		DB	0	;EXM
E432 2B01		DW	299	;DSM
E434 7F00		DW	127	;DRM
E436 C0		DB	0C0H	;AL0
E437 00		DB	0	;AL1
E438 2000		DW	32	;CKS
E43A 0200		DW	2	;OFF
E43C 74		DB	74H	;16*(#CPM SECTORS/PHYSICAL SECTOR) -1) + ;LOG2(#BYTES PER SECTOR/128) + 1 + ;8 IF DOUBLE SIDED.

```
*****
* THE FOLLOWING DPB DEFINES A DISKETTE FOR 128 BYTE SECTORS,
* SINGLE DENSITY, AND DOUBLE SIDED.
*****
```

E43D 3400	DPB128D	DW	52	;CP/M SECTORS/TRACK
E43F 04		DB	4	;BSH
E440 0F		DB	15	;BLM
E441 01		DB	1	;EXM
E442 F200		DW	242	;DSM
E444 7F00		DW	127	;DRM
E446 C0		DB	0C0H	;AL0
E447 00		DB	0	;AL1
E448 2000		DW	32	;CKS
E44A 0200		DW	2	;OFF
E44C 09		DB	9H	

```
*****
* THE FOLLOWING DPB DEFINES A DISKETTE AS 256 BYTE SECTORS,
* DOUBLE DENSITY, AND DOUBLE SIDED.
*****
```

E44D 6800	DPB256D	DW	104	;CP/M SECTORS/TRACK
E44F 04		DB	4	;BSH
E450 0F		DB	15	;BLM
E451 00		DB	0	;EXM
E452 E601		DW	486	;DSM
E454 FF00		DW	255	;DRM
E456 F0		DB	0F0H	;AL0

E457 00 DB 0 ;ALL  
E458 4000 DW 64 ;CKS  
E45A 0200 DW 2 ;OFF  
E45C 1A DB 1AH

\*\*\*\*\*  
\*  
\* THE FOLLOWING DPB DEFINES A DISKETTE AS 512 BYTE SECTORS,  
\* DOUBLE DENSITY, AND DOUBLE SIDED.  
\*  
\*\*\*\*\*

E45D 7800 DPB512D DW 120 ;CP/M SECTORS/TRACK  
E45F 04 DB 4 ;BSH  
E460 0F DB 15 ;BLM  
E461 00 DB 0 ;EXM  
E462 3102 DW 561 ;DSM  
E464 FF00 DW 255 ;DRM  
E466 F0 DB 0F0H ;AL0  
E467 00 DB 0 ;AL1  
E468 4000 DW 64 ;CKS  
E46A 0200 DW 2 ;OFF  
E46C 3B DB 3BH

\*\*\*\*\*  
\*  
\* THE FOLLOWING DPB DEFINES A DISKETTE AS 1024 BYTE SECTORS,  
\* DOUBLE DENSITY, AND DOUBLE SIDED.  
\*  
\*\*\*\*\*

E46D 8000 DP1024D DW 128 ;CP/M SECTORS/TRACK  
E46F 04 DB 4 ;BSH  
E470 0F DB 15 ;BLM  
E471 00 DB 0 ;EXM  
E472 5702 DW 599 ;DSM  
E474 FF00 DW 255 ;DRM  
E476 F0 DB 0F0H ;AL0  
E477 00 DB 0 ;AL1  
E478 4000 DW 64 ;CKS  
E47A 0200 DW 2 ;OFF  
E47C 7C DB 7CH

ENDIF

\*\*\*\*\*  
\*  
\* THE FOLLOWING DPB DEFINES A 10 MEGABYTE HARD DISK, WITH 512  
\* BYTE SECTORS.  
\*  
\*\*\*\*\*

E47D 0004 DPBHD1 IF MAXHD NE 0  
IF M26 NE 0  
DW 1024 ;CP/M SECTORS/TRACK  
E47F 05 DB 5 ;BSH  
E480 1F DB 31 ;BLM

E481 01	DB	1	;EXM	
E482 B507	DW	1973	;DSM	
E484 FF01	DW	511	;DRM	
E486 FF	DB	0FFH	;ALØ	
E487 FF	DB	0FFH	;AL1	
E488 0000	DW	0	;CKS	
E48A 0100	DW	1	;OFF	
E48C 33	DB	33H	:16*((#CPM SECTORS/PHYSICAL SECTOR) -1) + :LOG2(#BYTES PER SECTOR/128) + 1 + :8 IF DOUBLE SIDED.	
E48D 0004	DPBHD2	DW	1024	:CP/M SECTORS/TRACK
E48F 05	DB	5	;BSH	
E490 1F	DB	31	;BLM	
E491 01	DB	1	;EXM	
E492 B507	DW	1973	;DSM	
E494 FF01	DW	511	;DRM	
E496 FF	DB	0FFH	;ALØ	
E497 FF	DB	0FFH	;AL1	
E498 0000	DW	0	;CKS	
E49A 4000	DW	64	;OFF	
E49C 33	DB	33H	:16*((#CPM SECTORS/PHYSICAL SECTOR) -1) + :LOG2(#BYTES PER SECTOR/128) + 1 + :8 IF DOUBLE SIDED.	
E49D 0004	DPBHD3	DW	1024	:CP/M SECTORS/TRACK
E49F 05	DB	5	;BSH	
E4A0 1F	DB	31	;BLM	
E4A1 01	DB	1	;EXM	
E4A2 B507	DW	1973	;DSM	
E4A4 FF01	DW	511	;DRM	
E4A6 FF	DB	0FFH	;ALØ	
E4A7 FF	DB	0FFH	;AL1	
E4A8 0000	DW	0	;CKS	
E4AA 7F00	DW	127	;OFF	
E4AC 33	DB	33H	:16*((#CPM SECTORS/PHYSICAL SECTOR) -1) + :LOG2(#BYTES PER SECTOR/128) + 1 + :8 IF DOUBLE SIDED.	
ENDIF				
DPBHD1	IF	M1Ø NE Ø		
DPBHD1	DW	336	:CP/M SECTORS/TRACK	
DPBHD1	DB	5	;BSH	
DPBHD1	DB	31	;BLM	
DPBHD1	DB	1	;EXM	
DPBHD1	DW	1269	;DSM	
DPBHD1	DW	511	;DRM	
DPBHD1	DB	0FFH	;ALØ	
DPBHD1	DB	0FFH	;AL1	
DPBHD1	DW	0	;CKS	
DPBHD1	DW	1	;OFF	
DPBHD1	DB	33H	:16*((#CPM SECTORS/PHYSICAL SECTOR) -1) + :LOG2(#BYTES PER SECTOR/128) + 1 + :8 IF DOUBLE SIDED.	
DPBHD2	DW	336	:CP/M SECTORS/TRACK	
DPBHD2	DB	5	;BSH	
DPBHD2	DB	31	;BLM	
DPBHD2	DB	1	;EXM	

```
DW 1280 ;DSM
DW 511 ;DRM
DB 0FFH ;AL0
DB 0FFH ;AL1
DW 0 ;CKS
DW 122 ;OFF
DB 33H ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
;LOG2(#BYTES PER SECTOR/128) + 1 +
;8 IF DOUBLE SIDED.

ENDIF
IF M20 NE 0
DPBHD1 DW 672 ;CP/M SECTORS/TRACK
DB 5 ;BSH
DB 31 ;BLM
DB 1 ;EXM
DW 2015 ;DSM
DW 511 ;DRM
DB 0FFH ;AL0
DB 0FFH ;AL1
DW 0 ;CKS
DW 1 ;OFF
DB 33H ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
;LOG2(#BYTES PER SECTOR/128) + 1 +
;8 IF DOUBLE SIDED.

DPBHD2 DW 672 ;CP/M SECTORS/TRACK
DB 5 ;BSH
DB 31 ;BLM
DB 1 ;EXM
DW 2015 ;DSM
DW 511 ;DRM
DB 0FFH ;AL0
DB 0FFH ;AL1
DW 0 ;CKS
DW 98 ;OFF
DB 33H ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
;LOG2(#BYTES PER SECTOR/128) + 1 +
;8 IF DOUBLE SIDED.

DPBHD3 DW 672 ;CP/M SECTORS/TRACK
DB 5 ;BSH
DB 31 ;BLM
DB 1 ;EXM
DW 1028 ;DSM
DW 511 ;DRM
DB 0FFH ;AL0
DB 0FFH ;AL1
DW 0 ;CKS
DW 195 ;OFF
DB 33H ;16*((#CPM SECTORS/PHYSICAL SECTOR) -1) +
;LOG2(#BYTES PER SECTOR/128) + 1 +
;8 IF DOUBLE SIDED.

ENDIF
ENDIF
```

\*\*\*\*\*

\* \* \*

\* CP/M DISK PARAMETER HEADERS, UNINITIALIZED.  
\*  
\*\*\*\*\*

HEADER	MACRO	ND, DPB	
	DW	Ø	; TRANSLATION TABLE FILLED IN LATE
	DW	Ø, Ø, Ø	; SCRATCH
	DW	DIRBUF	; DIRECTORY BUFFER
	DW	DPB	; DPB FILLED IN LATER
	DW	CSV&ND	; DIRECTORY CHECK VECTOR
	DW	ALV&ND	; ALLOCATION VECTOR
	ENDM		
E4AD = 0000 #	DPBASE	EQU \$	
	DN	SET Ø	
		IF FIRST	
		REPT MAXHD	; GENERATE HARD DISK DPH'S FOLLOWED
	DN	HEADER %DN, DPBHD1	; BY FLOPPY DPH'S
	DN	SET DN+1	
	DN	HEADER %DN, DPBHD2	
	DN	SET DN+1	
		IF (M26 NE Ø) OR (M2Ø NE Ø)	
		HEADER %DN, DPBHD3	
	DN	SET DN+1	
		ENDIF	
		ENDM	
		REPT MAXFLOP	
	DN	HEADER %DN, Ø	
	DN	SET DN+1	
		ENDM	
		ELSE	
		REPT MAXFLOP	; GENERATE FLOPPY DPH'S FOLLOWED BY
	DN	HEADER %DN, Ø	; HARD DISK DPH'S
	DN	SET DN+1	
	ENDM		
E4AD+0000	DW	Ø	; TRANSLATION TABLE FILLED IN LATE
E4AF+000000000000	DW	Ø, Ø, Ø	; SCRATCH
E4B5+07E9	DW	DIRBUF	; DIRECTORY BUFFER
E4B7+0000	DW	Ø	; DPB FILLED IN LATER
E4B9+D2E9	DW	CSVØ	; DIRECTORY CHECK VECTOR
E4BB+87E9	DW	ALVØ	; ALLOCATION VECTOR
E4BD+0000	DW	Ø	; TRANSLATION TABLE FILLED IN LATE
E4BF+000000000000	DW	Ø, Ø, Ø	; SCRATCH
E4C5+07E9	DW	DIRBUF	; DIRECTORY BUFFER
E4C7+0000	DW	Ø	; DPB FILLED IN LATER
E4C9+5DEA	DW	CSV1	; DIRECTORY CHECK VECTOR
E4CB+12EA	DW	ALV1	; ALLOCATION VECTOR
	REPT MAXHD		
	DN	HEADER %DN, DPBHD1	
	DN	SET DN+1	
	DN	HEADER %DN, DPBHD2	
	DN	SET DN+1	
		IF (M26 NE Ø) OR (M2Ø NE Ø)	
		HEADER %DN, DPBHD3	
	DN	SET DN+1	
		ENDIF	

```

        ENDM

E4CD+0000    DW    0          ;TRANSLATION TABLE FILLED IN LATER
E4CF+0000000000 DW    0,0,0    ;SCRATCH
E4D5+07E9    DW    DIRBUF    ;DIRECTORY BUFFER
E4D7+7DE4    DW    DPBHD1    ;DPB FILLED IN LATER
E4D9+94EB    DW    CSV2      ;DIRECTORY CHECK VECTOR
E4DB+9DEA    DW    ALV2      ;ALLOCATION VECTOR
E4DD+0000    DW    0          ;TRANSLATION TABLE FILLED IN LATER
E4DF+0000000000 DW    0,0,0    ;SCRATCH
E4E5+07E9    DW    DIRBUF    ;DIRECTORY BUFFER
E4E7+8DE4    DW    DPBHD2    ;DPB FILLED IN LATER
E4E9+8BEC    DW    CSV3      ;DIRECTORY CHECK VECTOR
E4EB+94EB    DW    ALV3      ;ALLOCATION VECTOR
E4ED+0000    DW    0          ;TRANSLATION TABLE FILLED IN LATER
E4EF+0000000000 DW    0,0,0    ;SCRATCH
E4F5+07E9    DW    DIRBUF    ;DIRECTORY BUFFER
E4F7+9DE4    DW    DPBHD3    ;DPB FILLED IN LATER
E4F9+82ED    DW    CSV4      ;DIRECTORY CHECK VECTOR
E4FB+8BEC    DW    ALV4      ;ALLOCATION VECTOR
        ENDIF

```

```

*****
*          *
*  CBIOS RAM LOCATIONS THAT DON'T NEED INITIALIZATION.  *
*          *
*****
```

```

E4FD 0000    CPMSEC  DW    0          ;CP/M SECTOR #
E4FF 00      CPMDRV  DB    0          ;CP/M DRIVE #
E500 00      CPMTRK  DB    0          ;CP/M TRACK #
E501 0000    TRUESEC DW    0          ;DISK JOCKEY SECTOR THAT CONTAINS CP/M SECTOR
E503 00      BUFDRV   DB    0          ;DRIVE THAT BUFFER BELONGS TO
E504 00      BUFTRK   DB    0          ;TRACK THAT BUFFER BELONGS TO
E505 0000    BUFSEC   DW    0          ;SECTOR THAT BUFFER BELONGS TO
E507 =      BUFFER   EQU   $         

*****
```

```

*          *
*  SIGNON MESSAGE OUTPUT DURING COLD BOOT.  *
*          *
*****
```

```

HEXNUM  MACRO  NUM
        IF      (NUM/16) > 9
        DB      (NUM/16 AND 0FH) + 'A' - 10
        ELSE
        DB      (NUM/16 AND 0FH) + '0'
        ENDIF
        IF      (NUM AND 0FH) > 9
        DB      (NUM AND 0FH) + 'A' - 10
        ELSE
        DB      (NUM AND 0FH) + "0"
        ENDIF
        ENDM

```

```

E507 0D0A0A    PROMPT DB    ACR,ALF,ALF

```

CP/M MACRO ASSEM 2.0 #042 \*\*\* Cbios For CP/M Ver. 2.2 \*\*\*

E50A 4D6F72726F DB 'Morrow Designs '  
E519 36 DB '0'+MSIZE/10 ;CP/M MEMORY SIZE  
E51A 30 DB '0'+(MSIZE MOD 10)  
E51B 4B2043502F DB 'K CP/M ' ;CP/M VERSION NUMBER  
E522 32 DB CPMREV/10+'0'  
E523 2E DB ''  
E524 32 DB (CPMREV MOD 10)+'0'  
E525 2C20436269 DB ', Cbios rev '  
E531 322E DB REVNUM/10+'0', '..' ;CBIOS REVISION NUMBER  
E533 38 DB REVNUM MOD 10+'0'  
IF MAXHD NE 0  
E534 2E DB ''  
E535 32 DB MREV/10+'0'  
E536 36 DB MREV MOD 10+'0'  
IF (M10 OR M20) AND SDELAY  
DB 'M'  
ENDIF  
IF (M10 OR M20) AND NOT SDELAY  
DB 'F'  
ENDIF  
ENDIF  
E537 0D0A DB ACR, ALF  
E539 466F7220 DB 'For '  
  
E53D 6120446973 IF MAXFLOP NE 0  
DB 'a Disk Jockey 2D @ '  
HEXNUM % (ORIGIN/256)  
E550+46 DB (240/16 AND 0FH) + 'A' - 10  
E551+30 DB (240 AND 0FH) + '0'  
E552 30304820 DB '00H '  
ENDIF  
  
E556 616E6420 IF (MAXHD NE 0) AND (MAXFLOP NE 0)  
DB 'and '  
ENDIF  
  
E55A 616E20 IF MAXHD NE 0  
IF MAXHD EQ 1  
DB 'an '  
ENDIF  
IF MAXHD EQ 2  
DB 'two '  
ENDIF  
IF MAXHD EQ 3  
DB 'three '  
ENDIF  
IF MAXHD EQ 4  
DB 'four '  
ENDIF  
IF MREV EQ 10  
DB 'M10 '  
ENDIF  
IF MREV EQ 20  
DB 'M20 '  
ENDIF  
IF MREV EQ 26

```

E55D 4D323620      DB      'M26 '
E561 6861726420      DB      'hard disk'
IF      MAXHD NE 1
DB      's'
ENDIF
E56A 204020      DB      '@'
HEXNUM  %HDORG
E56D+35      DB      (80/16 AND 0FH) + '0'
E56E+30      DB      (80 AND 0FH) + '0'
E56F 482E      DB      'H.'
ENDIF
E571 0D0A      DB      ACR,ALF
E573 0D0A      DB      ACR,ALF
E575 2020202020      DB      'THE W6GO/K6HHD LIST'
E58F 0D0A      DB      ACR,ALF
E591 2020202020      DB      'Electronics Enterprises'
E5AD 0D0A      DB      ACR,ALF
E5AF 2020202020      DB      'Rio Linda, California'
E5CA 0D0A      DB      ACR,ALF
E5CC 00      DB      0

```

```

*****
*   * UTILITY ROUTINE TO OUTPUT THE MESSAGE POINTED AT BY H&L,
*   * TERMINATED WITH A NULL.
*   *
*****
```

```

E5CD 7E      MESSAGE MOV  A,M      ;GET A CHARACTER OF THE MESSAGE
E5CE 23      INX   H      ;BUMP TEXT POINTER
E5CF A7      ANA   A      ;TEST FOR END
E5D0 C8      RZ    H      ;RETURN IF DONE
E5D1 E5      PUSH  H      ;SAVE POINTER TO TEXT
E5D2 4F      MOV   C,A      ;OUTPUT CHARACTER IN C
E5D3 CD0CDD  CALL  COUT     ;OUTPUT THE CHARACTER
E5D6 E1      POP   H      ;RESTORE THE POINTER
E5D7 C3CDE5  JMP   MESSAGE    ;CONTINUE UNTIL NULL REACHED

```

```

*****
*   * CBOOT IS THE COLD BOOT LOADER. ALL OF CP/M HAS BEEN LOADED IN
*   * WHEN CONTROL IS PASSED HERE.
*   *
*****
```

```

E5DA 310001  CBOOT  LXI  SP,TPA      ;SET UP STACK
E5DD 3EC0      MVI   A,INTIOBY
E5DF 320300  STA   IOBYTE
E5E2 CD2DDE  CALL  TINIT      ;INITIALIZE THE TERMINAL
E5E5 2107E5  LXI   H,PROMPT    ;PREP FOR SENDING SIGNON MESSAGE
E5E8 CDCDE5  CALL  MESSAGE     ;SEND THE PROMPT
E5EB AF      XRA   A      ;SELECT DISK A
E5EC 32FFE4  STA   CPMDRV

```

```

E5EF 320400      STA      CDISK
                  IF      (MAXFLOP NE 0) AND FIRST
                  STA      FLOPFLG
ENDIF
E5F2 2103DD      LXI      H, BIOS+3
E5F5 2201DD      SHLD    BIOS+1
E5F8 C358DE      JMP     GOCPM

E5FB              DS      512-($-BUFFER) ;MAXIMUM SIZE BUFFER FOR 512 BYTE SECTORS

E707              IF      MAXFLOP NE 0
                  DS      512          ;ADDITIONAL SPACE FOR FLOPPIES 1K SECTORS
ENDIF

E907              DIRBUF IF      (MAXFLOP NE 0) OR (MAXHD NE 0)
                  DS      128          ;DIRECTORY BUFFER
ENDIF

ALLOC  MACRO ND, AL, CS
ALV&ND DS   AL
CSV&ND DS   CS
ENDM

0000 #             DN      SET   0
                  IF      NOT FIRST
                  REPT   MAXFLOP
                  ALLOC  %DN, 75, 64
DN      SET   DN+1
ENDM

E987+             ALV0   DS   75
E9D2+             CSV0   DS   64
EA12+             ALV1   DS   75
EA5D+             CSV1   DS   64
                  REPT   MAXHD
                  IF      M26 NE 0
                  ALLOC  %DN, 247, 0
DN      SET   DN+1
                  ALLOC  %DN, 247, 0
DN      SET   DN+1
                  ALLOC  %DN, 247, 0
DN      SET   DN+1
ENDIF
                  IF      M10 NE 0
                  ALLOC  %DN, 159, 0
DN      SET   DN+1
                  ALLOC  %DN, 161, 0
DN      SET   DN+1
ENDIF
                  IF      M20 NE 0
                  ALLOC  %DN, 252, 0
DN      SET   DN+1
                  ALLOC  %DN, 252, 0
DN      SET   DN+1
                  ALLOC  %DN, 129, 0

```

```
DN      SET      DN+1
      ENDIF
      ENDM
EA9D+  ALV2    DS      247
EB94+  CSV2    DS      0
EB94+  ALV3    DS      247
EC8B+  CSV3    DS      0
EC8B+  ALV4    DS      247
ED82+  CSV4    DS      0

      ELSE

      REPT    MAXHD
      IF      M26 NE 0
      ALLOC  %DN,247,0
DN      SET      DN+1
      ALLOC  %DN,247,0
DN      SET      DN+1
      ALLOC  %DN,247,0
DN      SET      DN+1
      ENDIF
      IF      M10 NE 0
      ALLOC  %DN,159,0
DN      SET      DN+1
      ALLOC  %DN,161,0
DN      SET      DN+1
      ENDIF
      IF      M20 NE 0
      ALLOC  %DN,252,0
DN      SET      DN+1
      ALLOC  %DN,252,0
DN      SET      DN+1
      ALLOC  %DN,129,0
DN      SET      DN+1
      ENDIF
      ENDM
      REPT    MAXFLOP
      ALLOC  %DN,75,64
DN      SET      DN+1
      ENDM
      ENDIF
      END
ED82
```

0006 AACK	E22F ACCOK	000D ACR	0003 AETX	000A ALF
E987 ALV0	EA12 ALV1	EA9D ALV2	EB94 ALV3	EC8B ALV4
DEAB AUTOFLG	CF00 BDOS	A000 BIAS	DD00 BIOS	E503 BUFDRV
0080 BUFF	E507 BUFFER	E505 BUFSEC	E504 BUFRK	E10D BUFWRTN
E323 BUILD	E5DA CBOOT	C700 CCP	0004 CDISK	DE08 CICRT
DE08 CIPTR	DD88 CITBLE	F003 CITY	DDF3 CIUC1	DE08 CIUR1
DE08 CIUR2	DEA3 CLDBOT	DE8E CLDCMND	001A CLEAR	DE3B COCRT
DEAC COLDBEG	DEBA COLDEND	DDC9 COLPT	0004 COMPLT	DD42 CONIN
DD48 CONIN1	DD57 CONOUT	DD36 CONST	DDC9 COPTP	DE46 COPTR
DE4D COPTR1	DD90 COTBLE	F006 COTTY	DDD4 COUL1	DDF2 COUNT
DDC9 COUP1	DDC9 COUP2	DD0C COUT	E0ED CPMDMA	E4FF CPMDRV
0016 CPMREV	E4FD CPMSEC	E500 CPMTRK	DE1C CSCRT	DE1C CSPTR
DD3C CSREADR	DDB8 CSRTBLE	DDB0 CSTBLE	DE14 CSTTY	DDFF CSUC1
DE1C CSUR1	DE1C CSUR2	E9D2 CSV0	EA5D CSV1	EB94 CSV2
EC8B CSV3	ED82 CSV4	DEAA CWFLG	0008 DBLSID	E066 DCRC
E1E1 DECIDE	E1DD DECIDGO	E20F DELAY	E212 DELOOP	E907 DIRBUF
E0B9 DIVDONE	E068 DIVLOG	E06A DIVLOGX	E0AB DIVLOOP	F400 DJBOOT
F003 DJCIN	F006 DJCOUT	F42D DJDEN	F412 DJDMA	DD33 DJDRV
F42A DJERR	F409 DJHOME	F400 DJRAM	F415 DJREAD	F40F DJSEC
F41B DJSEL	F430 DJSIDE	F427 DJSTAT	F40C DJTRK	F021 DJTSTAT
F418 DJWRITE	DF4A DONOP	E46D DP1024D	E42D DP1024S	E43D DPB128D
E3FD DPB128S	E44D DPB256D	E40D DPB256S	E45D DPB512D	E41D DPB512S
E4AD DPBASE	E47D DPBHD1	E48D DPBHD2	E49D DPBHD3	E32E DRIVES
E001 DRVHD	E318 DRVPTR	0020 DRVRDY	0007 DSKCLK	E0C8 DTSLOP
0005 ENTRY	E19E FILL	0000 FIRST	DFAA FLOPOK	E10C FLUSH
E1C6 FREAD	E071 GETDPB	E262 GETSPT	DE58 GOCPM	E27F HDADD
0051 HDCMND	0050 HDCNTL	0053 HDDATA	E32A HDDISK	E242 HDDMA
E1E9 HDDRV	0052 HDFUNC	E1FA HDHOME	0050 HDORG	E2E3 HDPREP
E265 HDREAD	0051 HDRESLT	0004 HDRLEN	E250 HDSEC	E308 HDSECTR
E247 HDSIDE	0020 HDSPT	0050 HDSTAT	E21B HDTRK	E229 HDTRK2
E29A HDWRITE	E324 HEAD	DF51 HOME	0000 IDBUFF	0040 INDEX
E034 INDX1	E03B INDX2	00C0 INTIOBY	E0FA INTO	0003 IOBYTE
0008 ISBUFF	DD77 LIST	DD7A LIST1	DD82 LISTST	0003 LOGDSK
DE25 LSLPT	DDC0 LSTBLE	DD98 LTBLE	0000 M10	0000 M20
0001 M26	0002 MAXFLOP	0001 MAXHD	00F7 MDIR	E5CD MESSAGE
E0D7 MOVE	E1D2 MOVER	E1D4 MOVLOP	001A MREV	003C MSIZE
DF1E NEWDMA	DEFF NEWSEC	E12D NOADJST	DF16 NOWRAP	00FB NSTEP
00FC NULL	4A00 OFFSETC	0002 OPDONE	F000 ORIGIN	E0F5 OUTOF
DD72 PNCH1	E119 PREP	E2C9 PROCESS	E507 PROMPT	0004 PSTEP
DDA0 PTBLE	DD6C PUNCH	DDE9 PWAIT	E0F0 RDWR	DD62 READER
E099 READ	DD65 READERA	DD68 READR1	DE2A READY	E09D REDWRT
000A RETRIES	0002 RETRY	E122 RETRYLP	E17F RETRYOP	001C REVNUM
0001 RSECT	DDA8 RTBLE	E285 RTLOOP	0005 SCENBL	0001 SDELAY
0200 SECLEN	E0DC SECPSEC	E09E SECSIZ	DF58 SECTRAN	DD4C SELDEV
DF4B SETDMA	DF93 SETDRV	E049 SETDRV1	DF45 SETSEC	E210 SETTLE
DF53 SETTRK	DF6E SIDEA	DFEF SIDEOK	DF71 SIDEONE	DF77 SIDETWO
E233 SLOOP	DE17 STAT	E1FF STEPO	DFFF SUBFP	E01D TDELAY
DE2D TINIT	0001 TKZERO	0008 TMOUT	0100 TPA	DF60 TRANFP
DF8F TRANHD	E501 TRUESEC	DEBB WARBEG	DEBB WARMEND	DEFE WARMLOD
DF32 WARMRD	DD03 WBOOTE	DEBC WBOOT	0000 WBOT	000F WENABL
0010 WFAULT	DEC0 WFLG	000B WRESET	E092 WRITE	E104 WRITTYP
DF35 WRMREAD	E248 WSDONE	0005 WSECT	E2A6 WTLOOP	E3BC XLT124
E32F XLT128	E34A XLT256	E37F XLT512	E08A XLT5	E307 ZKEY
E062 ZRET				